List of Publications

1 M. Lajolo, C. Passerone, F. Bellifemine, A. Bonomo, G. Ghigo, P. Civera, and A. Sangiovanni-Vincentelli.
Hardware/Software Co-design for Image Processing
In *Proceeding of the IASTED International Conference on Signal Processing and Communications*,
pp. 242-245, Las Palmas de Gran Canaria (Spain), February 11-14, 1998.

2 J. Liu, M. Lajolo and A. Sangiovanni-Vincentelli.
Software Timing Analysis Using HW/SW Cosimulation and Instruction Set Simulator
In *Proceedings of the 6th IEEE International Workshop on Hardware/Software Codesign*,
pp. 65-69, Seattle, WA (U.S.A.), March 15-18, 1998.

3 M. Lajolo, A. Raghunathan, S. Dey, L. Lavagno and A. Sangiovanni-Vincentelli.
A case study on modeling shared memory access effects during performance analysis of HW/SW systems
In *Proceedings of the 6th IEEE International Workshop on Hardware/Software Codesign*,
pp. 117-121, Seattle, WA (U.S.A.), March 15-18, 1998.

4 M. Lajolo, L. Lavagno and A. Sangiovanni-Vincentelli.
Fast Instruction Cache Simulation Strategies in a Hardware/Software Co-Design Environment
In *Proceedings of the IEEE Asia and South Pacific Design Automation Conference (ASP-DAC'99)*,
pp. 347-350, Hong Kong, January 18-21, 1999.

5 M. Lajolo, A. Raghunathan, S. Dey, L. Lavagno and A. Sangiovanni-Vincentelli.
Efficient Power Estimation Techniques for HW/SW Systems
In *Proceedings of the IEEE VOLTA'99 International Workshop on Low Power Design* ,
pp. 191-199, Como, Italy, March 4-5, 1999.

6 M. Lajolo, M. Lazarescu and A. Sangiovanni-Vincentelli.
A Compilation-based Software Estimation Scheme for Hardware/Software Co-Simulation
In *Proceedings of the 7th IEEE International Workshop on Hardware/Software Codesign*,
pp. 85-89, Roma, Italy, May 3-5, 1999.

7 M. Lajolo, L. Lavagno, C. Passerone and A. Sangiovanni-Vincentelli.
A Parameter-based Mapping Scheme for Behavior/Architecture Co-Design
In *IEEE Workshop on Design, Test and Applications (WDTA'99)*,
pp. 37-40, Dubrovnik, Croatia, June 14-16, 1999.

8 M. Lajolo, L. Lavagno, and A. Sangiovanni-Vincentelli.
Fast Instruction Cache Simulation for Hardware/Software Co-Design
In *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences,*
Vol. E82-A, No. 11 November 1999, pp. 2475-2483.

9 M. Lajolo, L. Lavagno, M. Rebaudengo, M. Sonza Reorda, M. Violante.

Evaluating System Dependability in a Co-Design Framework
In *Proceedings of the IEEE DATE 2000*,
pp. 586-590, Paris, France, March 27-30, 2000.

10 M. Lajolo, A. Raghunathan, S. Dey, L. Lavagno.
Efficient Power Estimation Techniques for System-on-Chip Design
In *Proceedings of the IEEE DATE 2000*,
pp. 27-34, Paris, France, March 27-30, 2000.

11 M. Lajolo, L. Lavagno, M. Rebaudengo, M. Sonza Reorda, M. Violante.
Automatic Test Bench Generation for Simulation-based Validation
In *Proceedings of the 8th IEEE International Workshop on Hardware/Software Codesign*,
pp. 136-140, San Diego, CA (U.S.A.), May 3-5, 2000.

12 M. Lajolo, L. Lavagno, M. Rebaudengo, M. Sonza Reorda, M. Violante.
System-level Test Bench Generation in a Co-design Framework
In *Proceedings of the IEEE European Test Workshop 2000*,
pp. 23-26, Cascais, Portugal, May 23-26, 2000.

13 M. Lajolo, L. Lavagno, M. Sonza Reorda, M. Violante.
Early Power Estimation for System-On-Chip Designs
In *Proceedings of the IEEE PATMOS 2000*,
pp. 108-117, Gottingen, Germany, September 13-15, 2000.

14 M. Lajolo, L. Lavagno, M.Rebaudengo, M. Sonza Reorda, M. Violante.
Behavioral Test Vector Generation for System-on-Chip Designs
In *Proceedings of the IEEE High Level Design Validation and Test 2000*,
pp. 21-26, Berkeley, CA (U.S.A.), November 8-10, 2000.

15 M. Lazarescu, M. Lajolo, E. Harcourt, J. Bammi, L. Lavagno.
Compilation-based Software Performance Estimation for System Level Design
In *Proceedings of the IEEE High Level Design Validation and Test 2000*,
pp. 167-172, Berkeley, CA (U.S.A.), November 8-10, 2000.

16 M. Lajolo, M. Sonza Reorda, M. Violante.
Early evaluation of bus interconnects dependability for System-on-Chip Designs
In *Proceedings of the 14th IEEE International Conference on VLSI Design*,
pp. 371-376, Bangalore, India, January 3-7, 2001.

17 M. Lajolo, M. Sonza Reorda, M. Violante.
Exploring Test Solutions by means of System-level Design Tools
In *Proceedings of the XVI Conference on Design of Circuits and Integrated Systems (DCIS) 2001*,
Porto, Portugal, November 20-23, 2001.

18 M. Lajolo.
Bus Guardians: An Effective Solution for Online Detection and Correction of Faults Affecting System-On-Chip Buses
In *IEEE Transactions on VLSI Systems*,
Vol. 9, No. 6, December 2001, pp. 974 - 982.

19 D. Cavendish, M. Lajolo, H. Liu.
On the Support of Minimum Service Rates for Input Queue Switches
In *Proceedings of the IEEE International Conference on Communications (ICC) 2002,*
pp. 1315-1320, New York, NY (U.S.A), April 28 - May 2, 2002.

20 D. Cavendish, M. Lajolo, H. Liu.
On the Evaluation of Fairness for Input Queue Switches
In *Proceedings of the IEEE International Conference on Communications (ICC) 2002,*
pp. 996-1000, New York, NY (U.S.A), April 28 - May 2, 2002.

21 M. Lajolo, A. Raghunathan, S. Dey, L. Lavagno and A. Sangiovanni-Vincentelli.
Co-Simulation Based Power Estimation for System-on-Chip Design
In *IEEE Transactions on VLSI Systems,*
Vol. 10, No. 3, June 2002, pp. 253 - 266.

22 C. He, M. Lajolo, M. Jacome.
System-level Exploration of Queuing Management Schemes for Input Queue Packet Switches
In *Proceedings of IP Based SoC Design 2002*
pp. 233-238, Grenoble, France, October 30-31, 2002.

23 C. He, M. Lajolo, M. Jacome.
A Case Study of a System Level Approach to Exploration of Queuing Management Schemes for Input Queue Packet Switches
In *Proceedings of the 11th Euromicro Conference on Parallel, Distributed and Network based Processing*
Genova, Italy, February 5-7, 2003.

24 M. Lajolo, C. Passerone, L. Lavagno.
Scalable Techniques for System-level Co-Simulation and Co-Estimation
In *IEE Proceedings - Computers and Digital Techniques*
Vol. 150, No. 4, July 2003, pp. 227 - 238.

25 M. Lajolo.
Ip-Based SOC Design in a C-based design methodology
In *Proceedings of IP Based SoC Design 2003*
pp. 203-208, Grenoble, France, November 13-14, 2003.

26 C. He, M. Lajolo, M. Jacome.
Architectural Probing: A New Paradigm for Enabling Communication Refinement in SOC Design
In *IEE Proceedings - Computers and Digital Techniques*
Vol. 151, No. 1, January 2004, pp. 23 - 32.

27 M. Lajolo, M. Prevostini.
UML in an Electronic System Level Design Methodology
To be presented at *UML-SOC'04 - International Workshop on UML for SoC Design*
San Diego, CA (U.S.A), June 6, 2004.

Thesis

1 M. Lajolo.
Design and Synthesis of Programmable Components for ATM Networks.
Master's Thesis, Politecnico di Torino, February 1995.

2 M. Lajolo (Advisor: Prof. Luciano Lavagno).
Architectural Design and Performance Analysis of Embedded Systems.
Ph.D. Thesis, Politecnico di Torino, February 1999.

Marcello Lajolo

A compilation-based software estimation scheme for hardware/software co-simulation

Full text Pdf (437 KB)

Source **International Conference on Hardware Software Codesign** archive
Proceedings of the seventh international workshop on Hardware/software codesign table of contents
Rome, Italy
Pages: 85 - 89
Year of Publication: 1999
ISBN:1-58113-132-1

Authors Marcello Lajolo Politecnico di Torino, Turin, Italy
Mihai Lazarescu Politecnico di Torino, Turin, Italy
Alberto Sangiovanni-Vincentelli Univ. of California at Berkeley, Berkeley

Sponsors IEEE-CS : Computer Society
IFIP : International Federation for Information Processing
SIGSOFT: ACM Special Interest Group on Software Engineering
SIGDA: ACM Special Interest Group on Design Automation

Publisher ACM Press New York, NY, USA

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 Felice Balarin , Massimiliano Chiodo , Paolo Giusto , Harry Hsieh , Attila Jurecska , Luciano Lavagno , Claudio Passerone , Alberto Sangiovanni-Vincentelli , Ellen Sentovich , Kei Suzuki , Bassam Tabbara, Hardware-software co-design of embedded systems: the POLIS approach, Kluwer Academic Publishers, Norwell, MA, 1997

2 Kei Suzuki , Alberto Sangiovanni-Vincentelli, Efficient software performance estimation methods for hardware/software codesign, Proceedings of the 33rd annual conference on Design automation conference, p.605-610, June 03-07, 1996, Las Vegas, Nevada, United States

3 Jie Liu , Marcello Lajolo , Alberto Sangiovanni-Vincentelli, Software timing analysis using HW/SW cosimulation and instruction set simulator, Proceedings of the 6th international workshop on Hardware/software codesign, p.65-69, March 15-18, 1998, Seattle, Washington, United States

4 Mentor Graphics Seamless CVE Home Page. htw://vewve.memofg.eomtseamlessL

Examiner's Attempt to Locate date

5 Synopsys' Eagle Home Page, http:ttwww.synopsy.com.tw/produasthwsw[eatgle.f-h.html.

6 F. Stappert, "Predicting pipelining and caching behaviour of hard real-time programs:' 1998.

7 Vojin Živojnovic , Heinrich Meyr, Compiled HW/SW co-simulation, Proceedings of the 33rd annual conference on Design automation conference, p.690-695, June 03-07, 1996, Las Vegas, Nevada, United States

8 Sharad Malik , Margaret Martonosi , Yau-Tsun Steven Li, Static timing analysis of embedded software, Proceedings of the 34th annual conference on Design automation conference, p.147-152, June 09-13, 1997, Anaheim, California, United States

9 R. Ernst , W. Ye, Embedded program timing analysis based on path clustering and architecture classification, Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design, p.598-604, November 09-13, 1997, San Jose, California, United States

10 R.M. Stallraan, Using and Porring GNU CC. http://www.dd orie.com, tgnu/docs/gcc/gccaoc.html.

↑ CITINGS *3*

Jong-Yeol Lee , In-Cheol Park, Timed compiled-code simulation of embedded software for performance analysis of SOC design, Proceedings of the 39th conference on Design automation, June 10-14, 2002, New Orleans, Louisiana, USA

M. Meerwein , C. Baumgartner , T. Wieja , W. Glauert, Embedded systems verification with FGPA-enhanced in-circuit emulator, Proceedings of the 13th conference on International Symposium on System Synthesis, September 20-22, 2000, Madrid, Spain

Sungjoo Yoo , Gabriela Nicolescu , Iuliana Bacivarov , Wassim Youssef , Aimen Bouchhima , Ahmed A. Jerraya, Multi-level software validation for NOC, Networks on chip, Kluwer Academic Publishers, Hingham, MA, 2003

↑ INDEX TERMS

Primary Classification:
B. Hardware
↳ **B.1** CONTROL STRUCTURES AND MICROPROGRAMMING

Additional Classification:
I. Computing Methodologies
↳ **I.6** SIMULATION AND MODELING

General Terms:
Design, Theory

Keywords:
compilation, delay modeling, software estimation

↑ Collaborative Colleagues:

Marcello Lajolo:	Sujit Dey Luciano Lavagno Mihai Lazarescu Jie Liu Anand Raghunathan Matteo Sonza Reorda Alberto Sangiovanni-Vincentelli Massimo Violante			
Mihai Lazarescu:	Horst Bunke Gianpiero Cabodi Marcello Lajolo Luciano Lavagno Sergio Nocco Claudio Passerone Stefano Quer Alberto Sangiovanni-Vincentelli Andrew Turpin Svetha Venkatesh	Geoff A. W. West		
Alberto Sangiovanni-Vincentelli:	D. K. Arvind Adnan Aziz Felice Balarin Massimo Baleani Kaustav Banerjee Martin Baynes Mark Beardslee Gérard Berry Douglas Braun Robert Brayton Robert K. Brayton Jerry Burch Misha Burich Jeffrey Burns Raul Camposano Stefano Cardelli Erik Carlson Giorgio Casinovi Andrea Casotto Henry Chang Edoardo Charbon Rong Chen Massimilano Chiodo Massimiliano Chiodo Claudionor Coelho Ron Collett Nanette Collins Jordi Cortadella Tullio Cuatto Antonino Damiano Luca Daniel Srinivas Davadas	Masahiro Fujita Frank Gennari Ranjit Gharpurey Paolo Giusto Richard Goering Carlo Guardiani Roberto Guerrieri Paolo Guisto William Heller Dave Hightower Harry Hsieh Harry C. Hsieh Chenming Hu Jawahar Jain Yunjian Jiang Attila Jurecska Timothy Kam Masamichi Kawarabayashi Kurt Keutzer Sunil P. Khatri Chunghee Kim Desmond Andrew Kirkpatrick Alex Kondratyev Phil Koopman Marcello Lajolo Jim Lansford Ulrich Lauther Luciano Lavagno Steve Law Mihai Lazarescu Edward Lee	Sharad Malik Maq Mannan Radu Marculescu Grant Martin Jonathan Martin Marc Massot Kartikeya Mayaram Patrick C. McGeer Rick McGeer Ken McMillan Amit Mehrotra Robert G. Meyer Robert S. Meyer Trevor Meyerowitz Giovanni De Micheli Paolo Miliozzi Sandra Moral Rajeev Murgai Amit Nandi Amit Narayan Arnit Narayan A. Richard Newton Yoshihito Nishizaki	Richard L. Rudell Jagesh V. Sanghavi A. Sangiovanni-Vincentelli Claudio Sansoè Steve Sapiro Larry Saunders Hamid Savoj Carl Sechen Ellen Sentovich Ellen M. Sentovich Marco Sgroi Henry Sheng Narendra Shenoy Thomas Shiple Thomas Robert Shiple Steven E. Shulz Mário J. Silva Kanwar Jit Singh Vigyan Singhal Kei Suzuki Abdallah Tabbara Bassam Tabbara Johan Van Ginderdeuren Iasson Vassiliou Tiziano Villa Yosinori Watanabe Donald Webber

Alper Demir
Sujit Dey
Stephen Edwards
Daniel Engels
Eric Felt
Jerry Fiddler
Edward A. Lee
Bill Lin
Jie Liu
Hi Keung Ma
Hi-Keung Ma
Enrico Malavasi
Abdul Aziz Malik
Arlindo L. Oliveira
Ralf H. J. M. Otten
Davide Pandini
Claudio Passerone
Roberto Passerone
Claudio Passeronge
Yatish Patel
Claudio Pinello
Jan Rabaey
Anand Raghunathan
Rajeev K. Ranjan
Stephen Ricca
Howard S. Rifkin
Fablo Romeo
James A. Rowson
Ruey-Sing Wei
Jacob White
Jacob K. White
Wayne Wolf
Hormoz Yaghutiel
Alexandre Yakovlev
Guang Yang
Naeem Zafar
Stefano Zanella

International Conference on Hardware Software Codesign

Archive

CODES+ISSS '03 **Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign & system synthesis**

CODES '02 **Proceedings of the tenth international symposium on Hardware/software codesign**

CODES '01 **Proceedings of the ninth international symposium on Hardware/software codesign**

CODES '00 **Proceedings of the eighth international workshop on Hardware/software codesign**

CODES '99 **Proceedings of the seventh international workshop on Hardware/software codesign**

CODES/CASHE '98 **Proceedings of the 6th international workshop on Hardware/software codesign**

CODES '97 **Proceedings of the 5th International Workshop on Hardware/Software Co-Design**

CODES '96 **Proceedings of the 4th International Workshop on Hardware/Software Co-Design**

CODES '94 **Proceedings of the 3rd international workshop on Hardware/software co-design**

International Conference on Hardware Software Codesign archive

Proceedings of the seventh international workshop on Hardware/software codesign
citation
1999, Rome, Italy *1999*

Table of Contents

Development of an optimizing compiler for a Fujitsu fixed-point digital signal processor
Sreeranga P. Rajan, Masahiro Fujita, Ashok Sudarsanam, Sharad Malik
Pages: 2 - 6
Full text available: Pdf(485 KB)
Additional Information: full citation, references, citings, index terms

Instruction set selection for ASIP design
Michael Gschwind
Pages: 7 - 11
Full text available: Pdf(501 KB)
Additional Information: full citation, references, index terms

Resource constrained dataflow retiming heuristics for VLIW ASIPs
M. Jacome, G. de Veciana, C. Akturan
Pages: 12 - 16
Full text available: Pdf(503 KB)
Additional Information: full citation, references, citings, index terms

An ASIP design methodology for embedded systems
Kayhan Küçükçakar
Pages: 17 - 21
Full text available: Pdf(385 KB)
Additional Information: full citation, references, citings, index terms

Automatic detection of recurring operation patterns
Marnix Arnold, Henk Corporaal
Pages: 22 - 26
Full text available: Pdf(399 KB)
Additional Information: full citation, references, citings, index terms

A flexible code generation framework for the design of application specific programmable processors
François Charot, Vincent Messé
Pages: 27 - 31
Full text available: Pdf(407 KB)
Additional Information: full citation, references, citings, index terms

An MPEG-2 decoder case study as a driver for a system level design methodology
Pieter van der Wolf, Paul Lieverse, Mudit Goel, David La Hei, Kees Vissers
Pages: 33 - 37

Full text available: Pdf(445 KB)
Additional Information: full citation, references, citings, index terms

Timed executable system specification of an ADSL modem using a C++ based design environment: a case study
Dirk Desmet, Michiel Esvelt, Probhat Avasare, Diederik Verkest, Hugo De Man
Pages: 38 - 42
Full text available: Pdf(438 KB)
Additional Information: full citation, references, citings, index terms

Flexible design of SPARC cores: a quantitative study
Tomás Bautista, Antonio Núñez
Pages: 43 - 47
Full text available: Pdf(468 KB)
Additional Information: full citation, references, index terms

Hardware/software co-design of an avionics communication protocol interface system: an industrial case study
François Clouté, Jean-Noël Contensou, Daniel Esteve, Pascal Pampagnin, Philippe Pons, Yves Favard
Pages: 48 - 52
Full text available: Pdf(337 KB)
Additional Information: full citation, references, citings, index terms

Multilanguage design of heterogeneous systems
P. Coste, F. Hessel, Ph. Le Marrec, Z. Sugar, M. Romdhani, R. Suescun, N. Zergainoh, A. A. Jarraya
Pages: 54 - 58
Full text available: Pdf(495 KB)
Additional Information: full citation, references, citings, index terms

The case for a configure-and-execute paradigm
Frank Vahid, Tony Givargis
Pages: 59 - 63
Full text available: Pdf(485 KB)
Additional Information: full citation, references, citings, index terms

Designing digital video systems: modeling and scheduling
H. J. H. N. Kenter, C. Passerone, W. J. M. Smits, Y. Watanabe, A. L. Sangiovanni-Vincentelli
Pages: 64 - 68
Full text available: Pdf(485 KB)
Additional Information: full citation, references, citings, index terms

Fast prototyping: a system design flow for fast design, prototyping and efficient IP reuse
Francois Pogodalla, Richard Hersemeule, Pierre Coulomb
Pages: 69 - 73
Full text available: Pdf(443 KB)
Additional Information: full citation, references, citings, index terms

Optimized rapid prototyping for real-time embedded heterogeneous multiprocessors
T. Grandpierre, C. Lavarenne, Y. Sorel
Pages: 74 - 78
Full text available: Pdf(516 KB)
Additional Information: full citation, references, citings, index terms

Using codesign techniques to support analog functionality
Francis G. Wolff, Michael J. Knieser, Dan J. Weyer, Chris A. Papachristou
Pages: 79 - 84
Full text available: Pdf(434 KB)
Additional Information: full citation, references, index terms

A compilation-based software estimation scheme for hardware/software co-simulation
Marcello Lajolo, Mihai Lazarescu, Alberto Sangiovanni-Vincentelli
Pages: 85 - 89

Full text available: Pdf(437 KB)
Additional Information: full citation, references, citings, index terms

A probabilistic performance metric for real-time system design
Tao Zhou, Xiaobo (Sharon) Hu, Edwin H.-M. Sha
Pages: 90 - 94
Full text available: Pdf(466 KB)
Additional Information: full citation, references, citings, index terms

Iterative cache simulation of embedded CPUs with trace stripping
Zhao Wu, Wayne Wolf
Pages: 95 - 99
Full text available: Pdf(468 KB)
Additional Information: full citation, references, citings, index terms

Optimizing geographically distributed timed cosimulation by hierarchically grouped messages
Sungjoo Yoo, Kiyoung Choi
Pages: 100 - 104
Full text available: Pdf(447 KB)
Additional Information: full citation, references, citings, index terms

Peer-based multithreaded executable co-specification
Donald E. Thomas, JoAnn M. Paul, Simon N. Peffers, Sandra J. Weber
Pages: 105 - 109
Full text available: Pdf(518 KB)
Additional Information: full citation, references, citings, index terms

Timing coverification of concurrent embedded real-time systems
Pao-Ann Hsiung
Pages: 110 - 114
Full text available: Pdf(486 KB)
Additional Information: full citation, references, citings, index terms

Worst-case analysis of discrete systems based on conditional abstractions
Felice Balarin
Pages: 115 - 119
Full text available: Pdf(355 KB)
Additional Information: full citation, references, citings, index terms

A unified formal model of ISA and FSMD
Jianwen Zhu, Daniel D. Gajski
Pages: 121 - 125
Full text available: Pdf(397 KB)
Additional Information: full citation, references, index terms

Co-design tool construction using APICES
Ansgar Bredenfeld
Pages: 126 - 130
Full text available: Pdf(399 KB)
Additional Information: full citation, references, index terms

Graph based communication analysis for hardware/software codesign
Peter Voigt Knudsen, Jan Madsen
Pages: 131 - 135
Full text available: Pdf(554 KB)
Additional Information: full citation, references, index terms

System synthesis utilizing a layered functional model
Ingo Sander, Axel Jantsch
Pages: 136 - 140
Full text available: Pdf(587 KB)

Additional Information: full citation, references, citings, index terms

Communication refinement in video systems on chip
J.-Y. Brunel, E. A. de Kock, W. M. Kruijtzer, H. J. H. N. Kenter, W. J. M. Smits
Pages: 142 - 146
Full text available: Pdf(406 KB)
Additional Information: full citation, references, citings, index terms

Compiling Esterel into sequential code
Stephen A. Edwards
Pages: 147 - 151
Full text available: Pdf(460 KB)
Additional Information: full citation, references, citings, index terms

Power estimation for architectural exploration of HW/SW communication on system-level buses
William Fornaciari, Donatella Sciuto, Christina Silvano
Pages: 152 - 156
Full text available: Pdf(492 KB)
Additional Information: full citation, references, citings, index terms

Software controlled power management
Yung-Hsiang Lu, Tajana Šimunić, Giovanni De Micheli
Pages: 157 - 161
Full text available: Pdf(351 KB)
Additional Information: full citation, references, citings, index terms

A statechart based HW/SW codesign system
I. D. Bates, E. G. Chester, D. J. Kinniment
Pages: 162 - 166
Full text available: Pdf(385 KB)
Additional Information: full citation, references, index terms

3D exploration of software schedules for DSP algorithms
J. Teich, E. Zitzler, S. S. Bhattacharyya
Pages: 168 - 172
Full text available: Pdf(580 KB)
Additional Information: full citation, references, index terms

Scheduling hardware/software systems using symbolic techniques
Karsten Strehl, Lothar Thiele, Dirk Ziegenbein, Rolf Ernst, Jürgen Teich
Pages: 173 - 177
Full text available: Pdf(534 KB)
Additional Information: full citation, references, citings, index terms

Scheduling with optimized communication for time-triggered embedded systems
Paul Pop, Petru Eles, Zebo Peng
Pages: 178 - 182
Full text available: Pdf(494 KB)
Additional Information: full citation, references, citings, index terms

A hardware-software cosynthesis technique based on heterogeneous multiprocessor scheduling
Hyunok Oh, Soonhoi Ha
Pages: 183 - 187
Full text available: Pdf(431 KB)
Additional Information: full citation, references, index terms

Embedded system synthesis under memory constraints
Jan Madsen, Peter Bjørn-Jørgensen
Pages: 188 - 192
Full text available: Pdf(389 KB)
Additional Information: full citation, references, citings, index terms

Overhead effects in real-time preemptive schedules
David L. Rhodes, Wayne Wolf
Pages: 193 - 197
Full text available: Pdf(484 KB)
Additional Information: full citation, references, index terms

System-level partitioning with uncertainty
Jones Albuquerque, Claudionor Coelho, Jr., Carlos Frederico Cavalcanti, Diógenes Cecilio da Silva, Jr., Antônio Otávio Fernandes
Pages: 198 - 202
Full text available: Pdf(432 KB)
Additional Information: full citation, references, index terms

Timing-driven HW/SW codesign based on task structuring and process timing simulation
Dinesh Ramanathan, Ali Dasdan, Rajesh Gupta
Pages: 203 - 207
Full text available: Pdf(543 KB)
Additional Information: full citation, references, citings, index terms

Aspects of system-level design
Jonas Plantin, Erik Stoy
Pages: 209 - 210
Full text available: Pdf(170 KB)
Additional Information: full citation, references, index terms

How standards will enable hardware/software co-design
Mark Genoe, Chris Lennard, Joachim Kunkel, Brian Bailey, Gjalt de Jong, Grant Martin, Kamal Hashmi, Shay Ben-Chorin, Anssi Haverinnen
Pages: 211 - 212
Full text available: Pdf(159 KB)
Additional Information: full citation, index terms

Published in **International Conference on Hardware Software Codesign**
Term used **performance**

Found **8** of **29**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 8 of 8

Relevance scale

1 **Case studies 1: Design flow for hardware/software cosynthesis of a video compression system**
Jörg Wilberg, Raul Camposano, Wolfgang Rosenstiel
September 1994 **Proceedings of the 3rd international workshop on Hardware/software co-design**
Full text available: pdf(690.42 KB) Additional Information: full citation, abstract, references

The implementation of a cosynthesis design flow in the CASTLE system is presented. The design flow generates a synthesizable hardware description and a C, C++, or Fortran compiler for an application-oriented processor. The approach is illustrated by the design of an embedded video compression system which can be integrated into the video card of a PC. The design flow is structured as follows: First, the requirements of the application programs are analyzed. Based on these analysis results, the d ...

2 **Case studies 2: A CoDesign experience with the MCSE methodology**
J. P. Calvez, D. Isidoro
September 1994 **Proceedings of the 3rd international workshop on Hardware/software co-design**
Full text available: pdf(686.14 KB) Additional Information: full citation, abstract, references

In this paper, we describe the experimentation of our codesign process to develop a communication system which needs to correctly mix hardware and software parts to satisfy required performance. The system design process is based on the MCSE methodology and we show its usefulness for CoDesign. CoDesign is shown as an enhancement of the implementation specification step of MCSE. System partitioning is the result of an interactive procedure based on performance and cost evaluations. The complete d ...

3 **Case studies 1: TigerSwitch: a case study in embedded computing system design**
Wayne Wolf, Andrew Wolfe, Steve Chinatti, Ravi Koshy, Gary Slater, Spencer Sun
September 1994 **Proceedings of the 3rd international workshop on Hardware/software co-design**
Full text available: pdf(689.04 KB) Additional Information: full citation, abstract, references

This paper describes and analyzes the design of TigerSwitch, a PC-based private branch exchange (PBX) designed at Princeton University. Building TigerSwitch required creating custom hardware and software designed to fit onto a standard IBM PC-compatible platform. Our design experience provides several lessons which we believe extend to other embedded

design domains: the system architecture required to meet performance goals is often not isomorphic to the structure of the specification; system-le ...

4 Analysis and synthesis: Automatic exploration of VLIW processor architectures from a designer's experience based specification
M. Auguin, F. Boeri, C. Carriere
September 1994 **Proceedings of the 3rd international workshop on Hardware/software co-design**
Full text available: pdf(581.94 KB) Additional Information: full citation, abstract, references

This paper presents a new synthesis approach for dedicated systems. The aim of our synthesis scheme is to achieve an automatic exploration of VLIW processor architectures from a pure C description of the input system. The innovation consists in the fact that unit allocation must manage the fact that a function may be realized either by dedicated functional units or by a set of lower-level efficiently controlled functional units. For example, execution of a square root function can be accomplishe ...

5 Hardware-software partitioning: Configuration-level hardware/software partitioning for real-time embedded systems
Joseph G. D'Ambrosio, Xiaobo (Sharon) Hu
September 1994 **Proceedings of the 3rd international workshop on Hardware/software co-design**
Full text available: pdf(706.42 KB) Additional Information: full citation, abstract, references

In this paper, we present an approach to hardware/software partitioning for real-time embedded systems. The abstraction level we have adopted is referred to as the configuration level, where hardware is modeled as resources with no detailed functionality and software is modeled as tasks utilizing the resources. Through configuration-level analysis, cost and performance tradeoffs can be studied early in the design process and a large design space can be explored. Feasibility factor is introduced ...

6 Verification: CASTLE: an interactive environment for HW-SW Co-Design
Markus Theißinger, Paul Stravers, Holger Veit
September 1994 **Proceedings of the 3rd international workshop on Hardware/software co-design**
Full text available: pdf(559.21 KB) Additional Information: full citation, abstract, references

We introduce CASTLE, a design environment for embedded systems. Starting from an algorithmic specification in C++/VHDL, CASTLE helps a designer to quickly find a suitable, cost-effective implementation of his system. The designer manually partitions the algorithmic specification into hardware and software components and refines the hardware architecture step by step. CASTLE provides immediate feed-back by displaying the feasibility and consequences of each partitioning decision. After partitioni ...

7 Models: Towards a declarative framework for hardware-software codesign
Wayne Luk, Teddy Wu
September 1994 **Proceedings of the 3rd international workshop on Hardware/software co-design**
Full text available: pdf(552.14 KB) Additional Information: full citation, abstract, references

We present an experimental framework for mapping declarative programs, written in a language known as Ruby, into various combinations of hardware and software. Strategies for parametrised partitioning into hardware and software can be captured concisely in this framework, and their validity can be checked using algebraic reasoning. The method has been used to guide the development of prototype compilers capable of producing, from a Ruby expression, a variety of implementations involving field-pr ...

8 Case studies 1: An example of applying the codesign method MOOSE

Peter Green, Paul Rushton, Ronnie Beggs

September 1994 **Proceedings of the 3rd international workshop on Hardware/software co-design**

Full text available: pdf(654.49 KB) Additional Information: full citation, abstract, references

An extended example of the application of a new method for the Object Oriented codesign of embedded systems (MOOSE) is presented. The example concerns on intelligent video system which is currently being developed using MOOSE. The paper highlights the notable features of the method (including executablity and the commitment process) with reference to the design of the video system, and presents tentative conclusions regarding the method's suitablity for embedded system codesign.

Results 1 - 8 of 8

Published in **International Conference on Hardware Software Codesign**
Term used **estimation**

Found **5** of **22**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 5 of 5

Relevance scale ■■■■■

1 A Co-Design Methodology Based on Formal Specification and High-level Estimation
C. Carreras, J. C. Lopez, M. L. Lopez, L. Sanchez, C. Delgado-Kloos, N. Martinez
March 1996 **Proceedings of the 4th International Workshop on Hardware/Software Co-Design**
Full text available: Publisher Site Additional Information: full citation, abstract

This paper presents a methodology for hardware-software co-design. It is based on the formal description technique LOTOS in the specification phase, and on estimation methods at different levels of abstraction in the partitioning phase. The LOTOS specification describes the system as a set of interacting communicating processes. Our HW-SW partitioning algorithm is guided by communications, performance and area estimates and by the suitability of each process for implementation in hardware or sof ...

Keywords: Co-design, formal specification, LOTOS, partitioning, estimation

2 The Interplay of Run-Time Estimation and Granularity in HW/SW Partitioning
Joerg Henkel, Rolf Ernst
March 1996 **Proceedings of the 4th International Workshop on Hardware/Software Co-Design**
Full text available: Publisher Site Additional Information: full citation, abstract

An important presupposition for HW/SW partitioning are sophisticated estimation algorithms at a high level of abstraction that obtain high quality results. Therefore the granularities of estimation and partitioning have to be adapted adequately. In this paper we discuss the effects that arise when the granularities of partitioning and estimation are not adapted in a necessary way. Furthermore we present our solution that allows to choose different levels of granularities adapted to the estimatio ...

3 Speed-up estimation for HW/SW-systems
W. Hardt, W. Rosenstiel
March 1996 **Proceedings of the 4th International Workshop on Hardware/Software Co-Design**
Full text available: Publisher Site Additional Information: full citation, abstract

HW/SW-codesign has been applied to a wide range of applications. Several partitioning

methods have been suggested. Thus the designer selects modules for HW or SW-implementation for the best possible performance within a set of performance and design constraints. This paper describes an estimation method to approximate a priori the entire system performance. The estimation method has been integrated into the codesign tool COD and first results could be generated. The estimated speed-up has been d ...

Keywords: HW/SW-systems, ciphering algorithm, codesign tool COD, hardware/software codesign, logic design, partitioning methods, speed-up estimation, system performance, systems analysis

4 PACE: A Dynamic Programming Algorithm for Hardware/Software Partitioning

Peter Voigt Knudsen, Jan Madsen

March 1996 **Proceedings of the 4th International Workshop on Hardware/Software Co-Design**

Full text available: Publisher Site Additional Information: full citation, abstract

This paper presents the PACE partitioning algorithm which is used in the LYCOS co-synthesis system for partitioning control/dataflow graphs into hardware- and software parts. The algorithm is a dynamic programming algorithm which solves both the problem of minimizing system execution time with a hardware area constraint and the problem of minimizing hardware area with a system execution time constraint. The target architecture consists of a single microprocessor and a single hardware chip (ASIC, ...

Keywords: Codesign, co-synthesis, hardware/software partitioning, communication, performance estimation, area estimation

5 Uninterpreted Co-Simulation for Performance Evaluation of Hw/Sw Systems

Jean Paul Calvez, Dominique Heller, Olivier Pasquier

March 1996 **Proceedings of the 4th International Workshop on Hardware/Software Co-Design**

Full text available: Publisher Site Additional Information: full citation, abstract

Performance modeling and evaluation of embedded hardware/software systems is important to help the CoDesign process. The hardware/software partitioning needs to be evaluated before synthesizing the solution. This paper presents a co-simulation technique based on the use of an uninterpreted model able to accurately represent the behavior of the whole system. The performance model includes two complementary viewpoints: the structural viewpoint which describes the functional structure, the hardware ...

Keywords: Hw/Sw systems, Performance evaluation, Co-Simulation, uninterpreted model

Results 1 - 5 of 5

Refine Search

Search Results -

Terms	Documents
L3 AND L6	54

Database:
- US Pre-Grant Publication Full-Text Database
- US Patents Full-Text Database
- US OCR Full-Text Database
- EPO Abstracts Database
- JPO Abstracts Database
- Derwent World Patents Index
- IBM Technical Disclosure Bulletins

Search: L8

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Wednesday, April 14, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=NO; OP=OR			
L8	L3 AND l6	54	L8
L7	L6 OR l3	8997	L7
L6	717/$$$.ccls.	4253	L6
L5	L4 and cosimulation	4	L5
L4	L3 AND simulation	1920	L4
L3	716/$$$.ccls.	4798	L3
L2	garbowski.XA. OR garbowski.XP.	442	L2
L1	garbowski.X$.	0	L1

END OF SEARCH HISTORY

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 4 of 4 returned.

☑ 1. Document ID: US 6691301 B2

L5: Entry 1 of 4 File: USPT Feb 10, 2004

US-PAT-NO: 6691301
DOCUMENT-IDENTIFIER: US 6691301 B2

TITLE: System, method and article of manufacture for signal constructs in a programming language capable of programming hardware architectures

DATE-ISSUED: February 10, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bowen; Matt	Littlemore			GB

US-CL-CURRENT: 717/114; 712/15, 716/16

ABSTRACT:

A system, method and article of manufacture are provided for using a dynamic object in a programming language. In general, an object is defined with an associated first value and second value. The first value is used in association with the object during a predetermined clock cycle. The second value is used in association with the object before or after the predetermined clock cycle.

18 Claims, 129 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 117

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Draw D

☑ 2. Document ID: US 6477683 B1

L5: Entry 2 of 4 File: USPT Nov 5, 2002

US-PAT-NO: 6477683
DOCUMENT-IDENTIFIER: US 6477683 B1

TITLE: Automated processor generation system for designing a configurable processor and method for the same

DATE-ISSUED: November 5, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Killian; Earl A.	Los Altos Hills	CA		
Gonzalez; Ricardo E.	Menlo Park	CA		
Dixit; Ashish B.	Mountain View	CA		
Lam; Monica	Menlo Park	CA		
Lichtenstein; Walter D.	Belmont	MA		
Rowen; Christopher	Santa Cruz	CA		
Ruttenberg; John C.	Newton	MA		
Wilson; Robert P.	Palo Alto	CA		
Wang; Albert Ren-Rui	Fremont	CA		
Maydan; Dror Eliezer	Palo Alto	CA		

US-CL-CURRENT: 716/1; 716/18

ABSTRACT:

An automated processor design tool uses a description of customized processor instruction set extensions in a standardized language to develop a configurable definition of a target instruction set, a Hardware Description Language description of circuitry necessary to implement the instruction set, and development tools such as a compiler, assembler, debugger and simulator which can be used to develop applications for the processor and to verify it. Implementation of the processor circuitry can be optimized for various criteria such as area, power consumption, speed and the like. Once a processor configuration is developed, it can be tested and inputs to the system modified to iteratively optimize the processor implementation. By providing a constrained domain of extensions and optimizations, the process can be automated to a high degree, thereby facilitating fast and reliable development.

104 Claims, 15 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 12

☑ 3. Document ID: US 6185518 B1

L5: Entry 3 of 4 File: USPT Feb 6, 2001

US-PAT-NO: 6185518
DOCUMENT-IDENTIFIER: US 6185518 B1

TITLE: Method and system for logic design constraint generation

DATE-ISSUED: February 6, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY

Chen; Liang T. Saratoga CA

US-CL-CURRENT: 703/19; 716/18, 716/6

ABSTRACT:

A system and method for generating design constraints for a logic synthesized block from timing analysis of the block. A timing analysis of logic described in software is performed for each of various operating modes of a circuit in which the logic is used. Timing data is extracted from the timing analysis and used as design constraints in the synthesis of the logic for the block.

27 Claims, 4 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 3

☑ 4. Document ID: US 6052524 A

L5: Entry 4 of 4 File: USPT Apr 18, 2000

US-PAT-NO: 6052524
DOCUMENT-IDENTIFIER: US 6052524 A

TITLE: System and method for simulation of integrated hardware and software components

DATE-ISSUED: April 18, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Pauna; Mark R.	Lombard	IL		

US-CL-CURRENT: 703/22; 703/19, 703/21, 703/28, 716/4

ABSTRACT:

A system and methods are provided to design, verify and develop simulated hardware and software components for a desired electrical device. The system includes a cycle-accurate simulator where X-number of simulator cycles is equivalent to Y-number of cycles on a simulated hardware component. The system further includes a simulator library for modeling and verifying hardware components of a desired electronic device. The simulator library includes built-in models and routines for simulating multiple internal hardware components. The simulator library is used with the cycle-accurate simulator. The system also includes a simulation Application Program Interface ("API") for allowing user-customized model and routines of internal and external hardware components to be used with the cycle-accurate simulator. The system can be used to design, verify and develop on-chip and off-chip components for a system-on-a-chip used in a desired electrical device. The methods provided include first method to simulate a requested operation for a simulated component far in the future with a fixed-length cycle counter by adjusting internal cycle counts in the cycle-accurate simulator for the requested operation and for the fixed-length cycle counter. A second method is used to send a

not-ready response for a requested operation, when the requested operation takes longer than an estimated number of cycles in the cycle-accurate simulator. The system and methods allow for detecting access errors, bus faults, invalid address translations, privilege protection violations, alignment violations and other timing and behavior violations for simulated hardware and software components and the integration thereof.

27 Claims, 6 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 6

Terms	Documents
L4 and cosimulation	4

G. Beltrame , C. Brandolese , W. Fornaciari , F. Salice , D. Sciuto , V. Trianni, Modeling assembly instruction timing in superscalar architectures, Proceedings of the 15th international symposium on System Synthesis, October 02-04, 2002, Kyoto, Japan

Marcello Lajolo , Anand Raghunathan , Sujit Dey, Efficient power co-estimation techniques for system-on-chip design, Proceedings of the conference on Design, automation and test in Europe, p.27-34, March 27-30, 2000, Paris, France

Marcello Lajolo , Anand Raghunathan , Sujit Dey , Luciano Lavagno, Cosimulation-based power estimation for system-on-chip design, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, v.10 n.3, p.253-266, June 2002

↑ INDEX TERMS

Primary Classification:

C. Computer Systems Organization

↳ **C.0** GENERAL

↳ **Subjects:** Instruction set design (e.g., RISC, CISC, VLIW)

Additional Classification:

C. Computer Systems Organization

↳ **C.0** GENERAL

↳ **Subjects:** Hardware/software interfaces

↳ **C.3** SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS

↳ **Subjects:** Real-time and embedded systems

↳ **C.4** PERFORMANCE OF SYSTEMS

↳ **Subjects:** Performance attributes

General Terms:

Algorithms, Design, Measurement, Performance

↑ Collaborative Colleagues:

Marcello Lajolo:	Sujit Dey	
	Luciano Lavagno	
	Mihai Lazarescu	
	Jie Liu	
	Anand Raghunathan	
	Matteo Sonza Reorda	
	Alberto Sangiovanni-Vincentelli	
	Massimo Violante	
Jie Liu:	Bob Broeg	David Olson
	Patrick Cheung	Vikram A. Saletore
	R. Currey	Alberto Sangiovanni-Vincentelli
	Leonidas Guibas	
	Marcello Lajolo	
	Yiu B. Lam	Feng Zhao

Edward A. Lee
Ted G. Lewis
Theodore G. Lewis
John Marsaglia

Hai Zhuge

Alberto Sangiovanni-Vincentelli:

D. K. Arvind
Adnan Aziz
Felice Balarin
Massimo Baleani
Kaustav Banerjee
Martin Baynes
Mark Beardslee
Gérard Berry
Douglas Braun
Robert Brayton
Robert K. Brayton
Jerry Burch
Misha Burich
Jeffrey Burns
Raul Camposano
Stefano Cardelli
Erik Carlson
Giorgio Casinovi
Andrea Casotto
Henry Chang
Edoardo Charbon
Rong Chen
Massimilano Chiodo
Massimiliano Chiodo
Claudionor Coelho
Ron Collett
Nanette Collins
Jordi Cortadella
Tullio Cuatto
Antonino Damiano
Luca Daniel
Srinivas Davadas
Alper Demir
Sujit Dey
Stephen Edwards
Daniel Engels
Eric Felt
Jerry Fiddler

Masahiro Fujita
Frank Gennari
Ranjit Gharpurey
Paolo Giusto
Richard Goering
Carlo Guardiani
Roberto Guerrieri
Paolo Guisto
William Heller
Dave Hightower
Harry Hsieh
Harry C. Hsieh
Chenming Hu
Jawahar Jain
Yunjian Jiang
Attila Jurecska
Timothy Kam
Masamichi Kawarabayashi
Kurt Keutzer
Sunil P. Khatri
Chunghee Kim
Desmond Andrew Kirkpatrick
Alex Kondratyev
Phil Koopman
Marcello Lajolo
Jim Lansford
Ulrich Lauther
Luciano Lavagno
Steve Law
Mihai Lazarescu
Edward Lee
Edward A. Lee
Bill Lin
Jie Liu
Hi Keung Ma
Hi-Keung Ma
Enrico Malavasi
Abdul Aziz Malik

Sharad Malik
Maq Mannan
Radu Marculescu
Grant Martin
Jonathan Martin
Marc Massot
Kartikeya Mayaram
Patrick C. McGeer
Rick McGeer
Ken McMillan
Amit Mehrotra
Robert G. Meyer
Robert S. Meyer
Trevor Meyerowitz
Giovanni De Micheli
Paolo Miliozzi
Sandra Moral
Rajeev Murgai
Amit Nandi
Amit Narayan
Arnit Narayan
A. Richard Newton
Yoshihito Nishizaki
Arlindo L. Oliveira
Ralf H. J. M. Otten
Davide Pandini
Claudio Passerone
Roberto Passerone
Claudio Passeronge
Yatish Patel
Claudio Pinello
Jan Rabaey
Anand Raghunathan
Rajeev K. Ranjan
Stephen Ricca
Howard S.

Richard L. Rudell
Jagesh V. Sanghavi
A. Sangiovanni-Vincentelli
Claudio Sansoè
Steve Sapiro
Larry Saunders
Hamid Savoj
Carl Sechen
Ellen Sentovich
Ellen M. Sentovich
Marco Sgroi
Henry Sheng
Narendra Shenoy
Thomas Shiple
Thomas Robert Shiple
Steven E. Shulz
Mário J. Silva
Kanwar Jit Singh
Vigyan Singhal
Kei Suzuki
Abdallah Tabbara
Bassam Tabbara
Johan Van Ginderdeuren
Iasson Vassiliou
Tiziano Villa
Yosinori Watanabe
Donald Webber
Ruey-Sing Wei
Jacob White
Jacob K. White
Wayne Wolf
Hormoz Yaghutiel
Alexandre Yakovlev
Guang Yang
Naeem Zafar
Stefano Zanella

Rifkin
Fablo Romeo
James A.
Rowson

↑ **Peer to Peer - Readers of this Article have also read:**

- Data structures for quadtree approximation and compression
  Communications of the ACM 28, 9
  Hanan Samet

- A hierarchical single-key-lock access control using the Chinese remainder theorem
  Proceedings of the 1992 ACM/SIGAPP Symposium on Applied computing
  Kim S. Lee , Huizhu Lu , D. D. Fisher

- 3D representations for software visualization
  Proceedings of the 2003 ACM symposium on Software visualization
  Andrian Marcus , Louis Feng , Jonathan I. Maletic

- Probabilistic surfaces: point based primitives to show surface uncertainty
  Proceedings of the conference on Visualization '02
  Gevorg Grigoryan , Penny Rheingans

- Efficient simplification of point-sampled surfaces
  Proceedings of the conference on Visualization '02
  Mark Pauly , Markus Gross , Leif P. Kobbelt

Software timing analysis using HW/SW cosimulation and instruction set simulator

Full text Publisher Site Pdf (38 KB)

Source **International Conference on Hardware Software Codesign** archive
Proceedings of the 6th international workshop on Hardware/software codesign table of contents
Seattle, Washington, United States
Pages: 65 - 69
Year of Publication: 1998
ISBN:0-8186-8442-9

Authors Jie Liu
Marcello Lajolo
Alberto Sangiovanni-Vincentelli

Sponsors SIGSOFT: ACM Special Interest Group on Software Engineering
SIGDA: ACM Special Interest Group on Design Automation
IEEE-CS : Computer Society
IFIP : International Federation for Information Processing

Publisher IEEE Computer Society Washington, DC, USA

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 Felice Balarin , Massimiliano Chiodo , Paolo Giusto , Harry Hsieh , Attila Jurecska , Luciano Lavagno , Claudio Passerone , Alberto Sangiovanni-Vincentelli , Ellen Sentovich , Kei Suzuki , Bassam Tabbara, Hardware-software co-design of embedded systems: the POLIS approach, Kluwer Academic Publishers, Norwell, MA, 1997

2 Ptolemy Home Page, "http://ptolemy.eecs.berkeley.edu"

3 Esterel Home Page, "http://www.inria.tr/meije/esterel"

4 R. Ernst , W. Ye, Embedded program timing analysis based on path clustering and architecture classification, Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design, p.598-604, November 09-13, 1997, San Jose, California, United States

5 K. ten Hagen, H. Meyr, "Timed and Untimed Hardware Software Cosimulation: Application and Efficient Implementation", Proceedings of Int. Workshop on Hardware-Software Codesign, Oct. 1993

6 E. A. Lee and A. Sangiovanni-Vincentelli, "A Denotational Framework for Comparing Models of

Computation", ERL Memorandum UCB/ERL M97/11, University of California, Berkeley, CA 94720, January 30, 1997.

7 S. Leef, "Hardware and Software Co-Verification - Key to Co- Design", Electronic Design, September 15,97, PP 67-7 1

8 C. Passerone, L.Lavagno, etc. "Trade-off Evaluation in Embedded System Design via Co-simulation". Proceedings of ASP-DAC, PP 291-297,1997.

9 Kurt Keutzer, Hardware/software co-simulation, Proceedings of the 31st annual conference on Design automation conference, p.439-440, June 06-10, 1994, San Diego, California, United States

10 Kei Suzuki , Alberto Sangiovanni-Vincentelli, Efficient software performance estimation methods for hardware/software codesign, Proceedings of the 33rd annual conference on Design automation conference, p.605-610, June 03-07, 1996, Las Vegas, Nevada, United States

11 "ST20 'Osprey' Toolset: User Manual", SGS-THOMSON Electronics, March 1997

12 S. Yoo, K. Choi, "Synchronization Overhead Reduction in Timed Cosimulation" Proceedings of Int. High Level Design Validation.

↑ CITINGS *11*

Marcello Lajolo , Mihai Lazarescu , Alberto Sangiovanni-Vincentelli, A compilation-based software estimation scheme for hardware/software co-simulation, Proceedings of the seventh international workshop on Hardware/software codesign, p.85-89, March 1999, Rome, Italy

M. Meerwein , C. Baumgartner , T. Wieja , W. Glauert, Embedded systems verification with FGPA-enhanced in-circuit emulator, Proceedings of the 13th conference on International Symposium on System Synthesis, September 20-22, 2000, Madrid, Spain

R. Ruelland , G. Gateau , T. Meynard , J. C. Hapiot, Digital emulator and observer of multicell converter, Mathematics and Computers in Simulation, v.63 n.3-5, p.335-347, 17 November 2003

Per Bjuréus , Axel Jantsch, Performance analysis with confidence intervals for embedded software processes, Proceedings of the international symposium on Systems synthesis, September 30-October 03, 2001, Montréal, P.Q., Canada

Benoit Clement , Richard Hersemeule , Etienne Lantreibecq , Bernard Ramanadin , Pierre Coulomb , Francois Pogodalla, Fast prototyping: a system design flow applied to a complex system-on-chip multiprocessor design, Proceedings of the 36th ACM/IEEE conference on Design automation conference, p.420-424, June 21-25, 1999, New Orleans, Louisiana, United States

J. Jung , S. Yoo , K. Choi, Performance improvement of multi-processor systems cosimulation based on SW analysis, Proceedings of the DATE 2001 on Design, automation and test in Europe, p.749-753, March 2001, Munich, Germany

Francois Pogodalla , Richard Hersemeule , Pierre Coulomb, Fast prototyping: a system design flow for fast design, prototyping and efficient IP reuse, Proceedings of the seventh international workshop on Hardware/software codesign, p.69-73, March 1999, Rome, Italy

Jeffry T. Russell , Margarida F. Jacome, Architecture-level performance evaluation of component-based embedded systems, Proceedings of the 40th conference on Design automation, June 02-06, 2003, Anaheim, CA, USA

Optimizing geographically distributed timed cosimulation by hierarchically grouped messages

Full text Pdf (447 KB)

Source **International Conference on Hardware Software Codesign** archive
Proceedings of the seventh international workshop on Hardware/software codesign table of contents
Rome, Italy
Pages: 100 - 104
Year of Publication: 1999
ISBN:1-58113-132-1

Authors Sungjoo Yoo Seoul National Univ., Seoul, Korea
Kiyoung Choi Seoul National Univ., Seoul, Korea

Sponsors IEEE-CS : Computer Society
IFIP : International Federation for Information Processing
SIGSOFT: ACM Special Interest Group on Software Engineering
SIGDA: ACM Special Interest Group on Design Automation

Publisher ACM Press New York, NY, USA

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 S. Yoo and K, Chol, "Synchronization Overhead Reduction in Timed Cosinmlafian", Proc. IEEE International High Level Design Validation and Test Workshop, pp. 157-164, Nov. 1997.

2 K. Hines and G. Bordello, "SelecAive Focus as a Means of Improvirlg Geographically Distributed Embedded Systeaa3 Co-simulation", Proc, Eighth IEEE International Workshop on Rapid System Prototyping, pp. 58--62, June 1997,

3 Ken Hines , Gaetano Borriello, A geographically distributed framework for embedded system design and validation, Proceedings of the 35th annual conference on Design automation conference, p.140-145, June 15-19, 1998, San Francisco, California, United States

4 Hassan Rajaei , Rassul Ayani , Lars-Erik Thorelli, The local Time Warp approach to parallel

simulation, Proceedings of the seventh workshop on Parallel and distributed simulation, p.119-126, May 16-19, 1993, San Diego, California, United States

5 Sungjoo Yoo , Kiyoung Choi, Optimistic distributed timed cosimulation based on thread simulation model, Proceedings of the 6th international workshop on Hardware/software codesign, p.71-75, March 15-18, 1998, Seattle, Washington, United States

6 D. Jaggar, AdvancdRISCMachinesArchitecturalRefernceManual, Prentice Hall. July 1996.

7 Teleuor, Telenar's H.263Soflware, htlp://www.nta.no/brukere/DVC/h263_ software/.

8 Pnr lable Video Research Group, PVRG-JPEG CODEC, ftp://havefimstanford .edu/pUbtjptg/JPEGv 1.2. I .lar.Z.

9 W. Jang, D. Lira, and S. Yon, ARM7 Instruction Set Simulator, http:ttpoppy.snu.ae.krfCodesigntARMISS/.

10 Synopsys, Inn., Synopsys Simulation Horaepage, htq:ftwww,synopsys.eoml produ ctsfsimulatlon/simulation.htnd.

11 D. Lim, K. Rha, and S. Yoo, Design Automation Lab. Protyping Board, http://poppy.snu.ac.kr/Codesign/DAL-P98A/.

↑ CITINGS *2*

J. Jung , S. Yoo , K. Choi, Performance improvement of multi-processor systems cosimulation based on SW analysis, Proceedings of the DATE 2001 on Design, automation and test in Europe, p.749-753, March 2001, Munich, Germany

Sungjoo Yoo , Jong-Eun Lee , Jinyong Jung , Kyungseok Rha , Youngchul Cho , Kiyoung Choi, Fast hardware-software coverification by optimistic execution of real processor, Proceedings of the conference on Design, automation and test in Europe, p.663-668, March 27-30, 2000, Paris, France

↑ INDEX TERMS

Primary Classification:

D. Software

↳ **D.3** PROGRAMMING LANGUAGES

↳ **D.3.4** Processors

↳ **Subjects:** Optimization

Additional Classification:

I. Computing Methodologies

↳ **I.6** SIMULATION AND MODELING

General Terms:

Design, Experimentation, Performance, Theory

↑ Collaborative Colleagues:

Kiyoung Choi:	Taekyoon Ahn	Seongsoo Hong	Yongjoo Kim	Arturo Salz
	Robert Alverson	Sun Y Hwang	Jong-Eun Lee	Dongwan Shin
	Tom Blank	Sun Young Hwang	Jong-eun Lee	Youngsoo Shin
	Soo-IK Chae	Jinhwan Jeon	KiJong Lee	Soonhoi
	Soo-Ik Chae	Byoungil Jeong	Sunghyun Lee	Larry Soule
	Young-Hoon Chang	Byungil Jeong	Sungtaek Lim	Wonyong Sung
	Youngchul Cho	Jinyong Jung	Sanghun Park	Jae-Hee Won
	Junghwan Choi	Jun-Woo Kang	Kyoungseok Rha	Sungjoo Yoo
	Nikil Dutt	Daehong Kim	Kyungseok Rha	
	Nikil D. Dutt	Jihong Kim	Thomas Rokicki	
	Dong Sam Ha	Kyuseok Kim	Takayasu Sakurai	
Sungjoo Yoo:	Ahmed A.Jerraya	Seongsoo Hong	Gabriela Nicolescu	
	Iuliana Bacivarov	Jinhwan Jeon	Wander O.Cesário	
	Amer Baghdadi	Byoungil Jeong	Yanick Paviot	
	Aimen Bouchhima	Byungil Jeong	Kyoungseok Rha	
	Youngchul Cho	Ahmed A. Jerraya	Kyungseok Rha	
	Kiyoung Choi	Ahmed Amine Jerraya	Wassim Youssef	
	Mario Diaz-Nava	Jinyong Jung		
	Lovic Gauthier	Jong-Eun Lee		
	Patrice Gerin	Sunghyun Lee		
	Dong Sam Ha	Damien Lyonnard		

↑ **Peer to Peer - Readers of this Article have also read:**

- Presenting computer algorithm knowledge units in computer science curriculum
  The Journal of Computing in Small Colleges 16, 2
  S. Krishnaprasad

- Developing teamwork through experiential learning
  The Journal of Computing in Small Colleges 16, 2
  Courtney S. Ferguson , Shade K. Little , Marilyn K. McClelland

- A pilot study for developing a quantitative model for outcomes assessment in the computer science program at a Small University
  The Journal of Computing in Small Colleges 16, 2
  Robert O. Jarman , Sankara N. Sethuraman

- Design of C++ classes and functions to model non-game interfacing applications of the inputs of a joy port (tutorial presentation)
  The Journal of Computing in Small Colleges 16, 2
  Lee R. Clendenning

- Teaching information systems management: an interactive case approach that is portfolio oriented
  The Journal of Computing in Small Colleges 16, 2
  N. Faye Angel, Ph.D.

Optimizing Geographically Distributed Timed Cosimulation by Hierarchically Grouped Messages*

Sungjoo Yoo Kiyoung Choi

Design Automation Laboratory
School of Electrical Engineering
Seoul National University
Seoul 151-742, Korea
{ysj,kchoi}@poppy.snu.ac.kr

Abstract

This paper presents a concept called **hierarchically grouped message** *to improve the performance of geographically distributed timed cosimulation. In the proposed method, messages which are transferred between simulators in a short period of simulated time are hierarchically grouped into a physical message to reduce the number of rollbacks in optimistic simulation as well as the communication overhead of message transfer. Experiments show the efficiency of the proposed method in an internationally distributed cosimulation environment.*

1 Introduction

In geographically distributed cosimulation environments, designers can simulate a system which consists of various remotely located intellectual property (IP) blocks without requiring local copies of the IP blocks. IP providers and EDA vendors also have benefits of allowing their IP blocks and proprietary tools, e.g. high performance hardware emulators, to be accessed while protecting their intellectual property rights.

However, high communication overhead in geographically distributed cosimulation environments prevents designers from performing detailed timed cosimulation of communication intensive systems. The problem gets more serious when *interrupt* is used as the communication protocol in the system being designed, since hardware and software simulators should synchronize with each other (via slow communication over Internet) at every system clock tick to detect the occurrence of interrupt [1].

There have been few researches on optimizing geographically distributed timed cosimulation. As an optimization method, [2][3] present a method, called *selective focus*, which dynamically changes the abstraction levels of communication models to allow designers to trade off between performance and accuracy. Contrary to [2][3], we present an optimization method which preserves the accuracy of detailed cosimulation.

In this paper, we focus on geographically distributed timed cosimulation of systems having interrupt as one of communication protocols. By a geographically distributed cosimulation environment, we mean a network of workstations (or PC's) over Internet (or Wide Area Network). Basically, our approach to the reduction of simulator synchronization overhead (including communication overhead) is to apply optimistic simulation concept to geographically distributed timed cosimulation since optimistic simulation is advantageous especially when communication overhead is dominant [1][4][5].

However, since communication overhead is excessive in geographically distributed cosimulation environments, the performance gain obtained by applying conventional optimistic simulation methods can be limited. In applying optimistic simulation to geographically distributed timed cosimulation, the effects of such high communication overhead on the increase of cosimulation run-time are twofold: (1) excessive rollbacks, i.e. rollback overhead caused by the slow transfer of messages compared to the simulation execution as well as (2) the communication overhead itself. According to our experiments, optimistic simulation suffers from excessive rollbacks when intensive synchronization between simulators is performed in a short period of simulated time. It is because while messages are being transferred via slow communication over Internet, the optimistic simulator that is to receive the messages runs further into the future, which causes rollbacks in the receiving simulator. To reduce such excessive rollbacks and high communication overhead, we present a concept called *hierarchically grouped message* (HM) where messages transferred between simulators in a short period of simulated time are hierarchically grouped into a physical message.

This paper is organized as follows. In Section 2, we give a brief description of applying optimistic simulation to timed cosimulation. Section 3 explains our motivation. We present hierarchically grouped message concept in Section 4. We give experimental results in Section 5. Section 6 concludes this paper.

2 Background

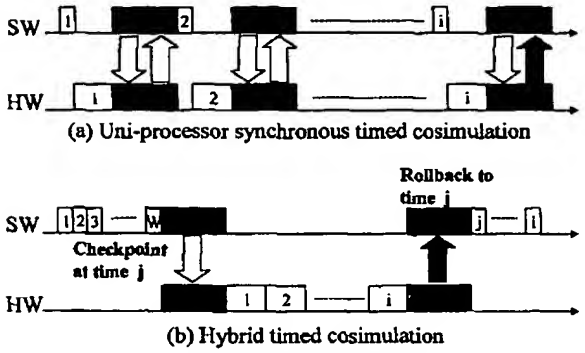
In this section, we describe three types of timed cosimulation considered in this paper: uni-processor synchronous cosimulation, hybrid cosimulation, optimistic distributed cosimulation. In the following, we define a *message* to be a timestamped event.

2.1 Uni-processor Synchronous Cosimulation

In Figure 1, we assume that software (SW) and hardware (HW) start to run concurrently at time 0. The exact time when HW sends an interrupt to SW is not known *a priori* but given as a time interval. In Figure 1, blank rectangles and numbers on them represent simulation workloads and the corresponding local times in the simulator they are running on, respectively. Blank arrows represent *null messages* for simulator synchronization only and shaded arrows represent interrupts from HW to SW. Shaded rectangles represent **simulator synchronization overhead** in cosimulation runtime. In *synchronous cosimulation* as shown in Figure 1 (a), SW

*This work was supported in part by ETRI, Korea.

CODES '99 Rome Italy



(a) Uni-processor synchronous timed cosimulation

(b) Hybrid timed cosimulation

Figure 1: Reduction of synchronization overhead by hybrid cosimulation.

and HW simulators synchronize with each other at every system clock tick to detect the occurrence of interrupt. In synchronous cosimulation, most of simulation run-time can be consumed in simulator synchronization [1].

2.2 Hybrid Cosimulation

Most simulators do not support optimistic simulation features such as checkpoint (or state saving) and rollback. The same is true for emulators which can hardly support a cycle-accurate state saving feature. Therefore we consider a hybrid cosimulation where optimistic simulator(s) and synchronous simulator(s) (including emulators) co-exist. A representative case of the application of hybrid cosimulation is accessing a high performance HW emulator via Internet.

In *hybrid cosimulation* [1] as shown in Figure 1 (b), to reduce simulator synchronization overhead, we first run the optimistic simulator for a time window of predetermined size **W**. In this example, we assume that the SW simulator is an optimistic simulator. The optimistic simulator stops after the time window **W** elapses or at a time point **W'** (< **W**) when a message is sent to the synchronous simulator (in this example, the HW simulator), and waits for messages from the synchronous simulator. During the simulation, states of optimistic simulation are stored at checkpoints in preparation for the rollback. The synchronous simulator starts to run until the time point when the optimistic simulation stops. The synchronous simulation may stop earlier if the synchronous simulator sends a message to the optimistic simulator. In this case, since the timestamp of the message sent to the optimistic simulator is earlier than the time point when the optimistic simulator has stopped, the optimistic simulator rolls back to a checkpoint before (or equal to) the timestamp of the message. If there is no message from the synchronous simulator to the optimistic simulator, then the synchronous simulator stops at the time point **W** (or **W'**). After determining a new **W**, the optimistic simulator starts to run until **W**. Then, the cosimulation continues in this way. Note that in hybrid cosimulation a simulator stops its simulation when it sends a message to another simulator or after the time window **W** or **W'** elapses.

2.3 Optimistic Distributed Cosimulation

In optimistic distributed cosimulation, a set of *logical processes* (physically, optimistic simulators) execute concurrently and communicate by exchanging messages. A logical process (LP), as a unit of parallel simulation, consists of (1) the simulation model of the sub-system being simulated, (2) a state queue to store the states of the simulation model, (3) an input message queue for messages which arrive at the LP, and (4) an output message queue for messages which the LP sends to other LP's. Each LP has its own local time called *local virtual time* (LVT). Each LP works as follows. After advancing LVT, the LP looks up the input message queue to find an input message having a timestamp equal to LVT, processes the message, and advances its LVT. If there is any unprocessed input message which has a timestamp earlier than LVT (we call such a message a *straggler message*), the LP rolls back its LVT according to the timestamp of the straggler message, i.e. the state stored at the time point earlier than or equal to the timestamp of the straggler message is restored.

To support rollback, states are stored at checkpoints. To constrain the memory usage of simulation host for state saving, a *global virtual time* (GVT) is calculated. GVT is the minimum of timestamps of *in-transit messages*[1] and local virtual times of all LP's. States and messages having timestamps earlier than GVT can be removed from the state queue and the input/output message queues.[2] For more details on optimistic distributed cosimulation, refer to [5]. A representative case of applying optimistic distributed cosimulation is accessing the simulation models of IP blocks and performing their simulations via Internet.

3 Motivation

Grouping multiple messages into fewer numbers of physical messages gives faster transmission of messages than transmitting each message separately, since the communication overhead over Internet does not strictly depend on the sizes of messages being transferred, but rather strongly depends on the number of physical messages transferred

Grouping messages also has the advantage of reducing the number of rollbacks. Figure 2 illustrates an example of communication of messages between a SW simulator and a HW simulator in optimistic distributed timed cosimulation. In Figure 2, we assume that SW receives 64 data from HW. In SW processors, such a communication can be performed by executing memory load instructions (e.g. **LDR** or **LDM** instructions in ARM7 processor [6]). To receive each of the data, SW sends the address value to HW (event on the address bus). HW sends the datum corresponding to the received address value to SW (event on the data bus). In memory load (or store) instructions, the time gap between the event on the address bus and the event on the data bus is within a few clock cycles in the simulated time.

However, due to high communication overhead (e.g. at least a few milliseconds per message transfer) in geographically distributed cosimulation environments, when the datum requested by SW arrives at the SW simulator, the SW simulator (one which has millions cycles/sec performance on high performance workstations) may have proceeded further into the future in the simulated time. Such a straggler message causes rollback in the receiving optimistic simulator, in this example, the SW simulator. Figure 2 also illustrates rollbacks (upward arcs) caused by such straggler messages. As shown in Figure 2 (a), optimistic distributed timed cosimulation suffers from excessive rollbacks when intensive synchronization between simulators is performed in a short period of simulated time.

To reduce such excessive rollbacks, we use a *hierarchically grouped message* (HM) concept. Figure 2 (b) shows simulator synchronization using HM's. In this example, we group 64 messages transferring from SW to HW (HW to SW) into a single physical message **HM2hw** (**HM2sw**). In constructing a new physical message, we neither merge original events into a new event nor increase

[1] Messages which are in the communication channels between LP's, or not processed yet in input message queues. In our implementation, the Internet communication channel works as a FIFO queue.

[2] If there is no state stored at GVT, the state having the latest timestamp (but earlier than GVT) is kept in the state queue.

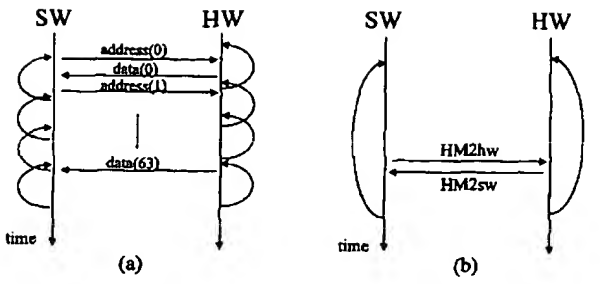


Figure 2: Reduction of rollbacks by hierarchically grouped messages.

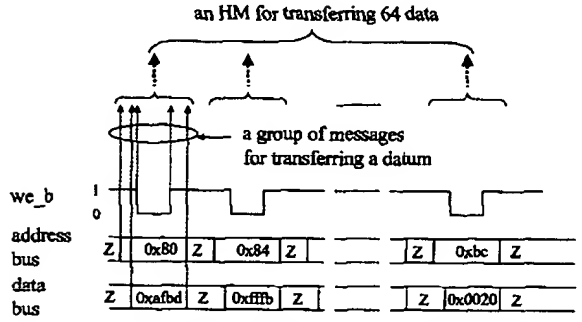


Figure 3: An example of hierarchically grouped message.

the abstraction levels of messages, i.e. original events (i.e. their abstraction levels) are kept unchanged in our method. Since only a single physical message is sent from SW (HW) to HW (SW) in Figure 2 (b), rollback occurs only twice in total.

Such reduction of excessive rollbacks, however, does not come for free. Since the transfer of messages is delayed for the construction of the whole HM, *rollback distance* (the amount of simulated time which is canceled by rollback) on each simulator may increase. In Section 5, however, we show experimentally that such a negative effect is negligible. Basically, since optimistic simulation is performed, the construction of HM's is possible. It is because the causality error caused by the delay of message transfer during the construction of HM's can be recovered by the rollback mechanism.

4 Hierarchically Grouped Messages

4.1 Specification of HM

For the explanation of specifying HM's, Figure 3 illustrates the construction of an HM for transferring 64 data from SW to HW. First, each message represents an event (or simultaneous events) on the address/data buses or control signals such as we_b (write enable bar). The transfer of each datum is specified as a group of messages as shown in Figure 3. The transfer of 64 data is specified as a group of groups of messages, each group of which represents the transfer of a datum. As such, higher level groups of messages are constructed by grouping lower level messages (or groups of messages) in a hierarchical way.

Each HM has an (or a set of) address range(s) associated with the data belonging to the HM. In Figure 3, the HM transferring 64 data has an address range from 0x80 to 0xbc. The designer can also specify an address range to construct an HM for the purpose of performance optimization.

4.2 Construction of HM during Simulation

During cosimulation, each simulator monitors the values on the address bus and starts to construct an HM by detecting the start address value (e.g. 0x80 in Figure 3) of the address range of the HM. During the construction of the HM, output messages are not sent to their receiving simulator. Instead, they are stored in the output message queue. If the simulator detects the end address of the address range (e.g. 0xbc in Figure 3), then the simulator creates a physical message with the unsent messages in the output message queue and sends it to the receiving simulator. We refer to the time period between the start time and the end time of the construction of an HM as an *HM construction period.*

From the implementational viewpoint, an HM is an array of messages. From the viewpoint of the receiving simulator which reads each incoming message one by one from the Internet communication channel, there is no difference between hierarchically grouped messages and separately sent messages. The construction of an HM requires proper modifications in the cases that (1) interrupt is allowed during the construction of an HM, (2) an HM is constructed during the data dependent execution, and (3) a synchronous simulator in hybrid cosimulation constructs an HM.

4.3 Handling Interrupts

Depending on whether interrupt is allowed during communication between SW and HW, we classify the hierarchically grouped message into two types: *interruptible* HM and *non-interruptible* HM. We define an interruptible HM as follows.

Definition 1 *If the execution of SW can be interrupted while SW is constructing (or processing messages belonging to) an HM, the HM is defined to be an interruptible HM.*

For example, while SW reads 64 data from HW, the execution of SW can be interrupted by a timer interrupt to the SW processor unless the interrupt is masked. For the non-interruptible HM, the execution of SW is guaranteed to be continued during the construction or reception of the HM. For the interruptible HM, the simulator sends a *partial* HM in the cases described below. By a partial HM, we mean an HM which has been constructed until some time point before the end address of the HM is reached.

For the interruptible HM, the simulator sends a partial HM in the following two cases.

Case 1 *While the HW simulator is constructing an interruptible HM, HW sends an interrupt to SW.*

In this case, since SW execution will be interrupted by the interrupt sent by HW, the transfer of the interruptible HM is not guaranteed to continue. Thus, the HW simulator stops constructing the HM and sends the partial HM to the SW simulator.

Case 2 *During the HM construction period of an interruptible HM, the SW simulator processes a message containing an interrupt event.*

In this case, since SW execution is interrupted by the interrupt event, the SW simulator sends the partial HM to the HW simulator.

4.4 Sending a Partial HM in Data Dependent Execution

To avoid large delay caused by the data dependent executions such as data dependent loops during the construction of an HM, the simulator sends the partial HM if the delay exceeds a given timeout value ($T_{timeout}$). That is, if LVT$-T_{HM_start} > T_{timeout}$, then the simulator sends the partial HM. T_{HM_start} represents the local virtual time when the simulator starts to construct the HM. The designer can set a timeout value $T_{timeout}$. If $T_{timeout} = 0$, then HM concept is not used.

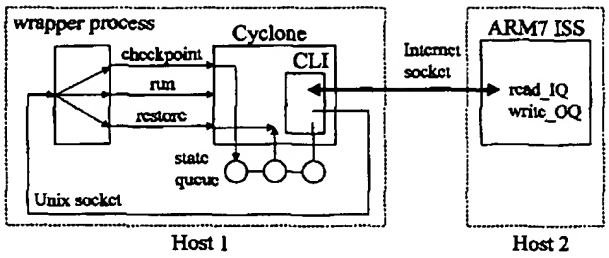


Figure 4: Optimistic distributed cosimulation using ARM7 ISS and Synopsys Cyclone.

4.5 Construction of HM in Hybrid Cosimulation

As explained in Section 2, in hybrid cosimulation the simulator stops its simulation when it sends a message to another simulator or after the time window W or W' elapses. However, in applying HM concept to hybrid cosimulation, the simulator does not stop its simulation during the construction of an HM. Therefore, it may continue the simulation beyond W or W'. After the construction of the HM, the simulator sends the HM to another simulator, stops its simulation, and waits for messages from the other simulator.

Basically, since HM concept is applied to optimistic simulation, only the optimistic simulator can construct HM's. For the non-interruptible HM, however, the synchronous simulator can also construct HM's in hybrid cosimulation since SW execution is guaranteed to be continued during the construction of the non-interruptible HM.

4.6 Calculation of GVT during the Construction of HM

In optimistic distributed cosimulation, when a simulator calculates GVT, it sends a request to the other simulators to obtain information for calculating GVT. When a simulator acknowledges to the request, it sends to the requesting simulator the minimum value of its LVT and the timestamps of unprocessed messages in its input message queue. When the simulator acknowledges to the request, if it is constructing an HM, then it sends to the requesting simulator the minimum value of its LVT, the timestamps of unprocessed input messages, and the timestamps of unsent output messages.

5 Experiments

We performed geographically distributed timed cosimulation for two examples : an H.263 decoder [7] and a JPEG encoder [8]. For the H.263 decoder, 3 frames of an image called Carphone (QCIF: 176x144 pixels) are decoded and for the JPEG encoder, a 116x96 image is encoded. For the HW parts of the examples, Discrete Cosine Transformation (DCT) and Inverse DCT functions are implemented. The other parts of the examples are implemented in SW.

We construct HM's for transferring 64 data from SW (HW) to HW (SW). In our implementation, a single original message has 44 bytes information. For transferring one datum from SW (HW) to HW (SW), four messages are transferred from the SW simulator to the HW simulator. In the case of transferring one datum from HW to SW, a single message is transferred from the HW simulator to the SW simulator together with four messages transferred from the SW simulator to the HW simulator. Thus, an HM from SW to HW contains 11,264 (=44x4x64) bytes and an HM from HW to SW contains 2,816 (=44x64) bytes in total.

We use an ARM7 instruction set simulator (ISS) having optimistic simulation features for SW simulation [9]. For optimistic HW simulation, we use a commercial cycle-based simulator, Synopsys Cyclone utilizing its checkpoint and restore functions [10]. Optimistic simulation library functions [5] are linked with ARM7 ISS and Cyclone, respectively. We also use a HW emulator [11] (based on Xilinx XC4085), which does not provide optimistic simulation features.

Table 1: Cosimulation run-times for the H.263 decoder.

N_{hop}	Opt. Dist. (sec)		Hybrid (sec)	
	w/o HM	w/ HM	w/o HM	w/ HM
3	3,727	2,436	4,579	406
12	5,900	4,200	74,577	1,457

Table 2: Cosimulation run-times for the JPEG encoder.

N_{hop}	Opt. Dist. (sec)		Hybrid (sec)	
	w/o HM	w/ HM	w/o HM	w/ HM
3	629	523	617	110
12	1,266	879	24,118	371

We use the number of *hops* N_{hop} to denote the number of Internet connections in a geographically distributed cosimulation environment.[3] To experiment the effect of communication overhead via Internet, we performed cosimulation in two different geographically distributed cosimulation environments (N_{hop} = 3 or 12). Especially, the case of N_{hop} = 12 is a connection between a workstation (or a PC) at Seoul Nat'l Univ. in Korea and a workstation at Virginia Tech. in the U.S.

For optimistic distributed cosimulation, we run ARM7 ISS and Cyclone on two remotely located simulation hosts (two SUN UltraSparc I's, 143 MHz). Figure 4 shows a simplified view of our optimistic distributed cosimulation. For the case of Synopsys Cyclone, we use C Language Interface (CLI) to link our optimistic simulation library functions with Cyclone. We also use a wrapper (a Unix process) to issue simulation commands (run, checkpoint, and restore as shown in Figure 4) to Cyclone. For hybrid cosimulation, we run ARM7 ISS (i.e. the optimistic simulator) on a workstation and the HW emulator (i.e. the synchronous simulator) on a PC (Pentium II, 300 MHz, Win98).

First, we ran uni-processor synchronous cosimulation of two examples using ARM7 ISS and Cyclone on an UltraSparc I workstation and obtained 5,816 sec (for H.263) and 1,418 sec (for JPEG) for the run-times. Table 1 and 2 show cosimulation run-times of two geographically distributed cosimulation environments. Compared to the run-times of uni-processor synchronous cosimulation, the performance improvement of optimistic distributed cosimulation (w/o HM) comes mainly from the reduction of simulator synchronization overhead rather than the benefit from parallel simulation. Applying HM concept to optimistic distributed cosimulation, we can obtain 1.53 and 1.40 times (1.20 and 1.44 times) performance improvement for the H.263 example (for the JPEG example) in the two cases of N_{hop}.

Table 3 shows the reduction of the numbers of rollbacks by applying HM concept to optimistic distributed cosimulation. Figure 5 shows the histograms of the numbers of rollbacks in the case of optimistic distributed cosimulation of the H.263 example (when N_{hop} = 12). In Figure 5, the number of short rollbacks is dramatically reduced by applying HM concept, while that of long rollbacks slightly increases due to the delay of message transfer caused by the

[3] In this paper, N_{hop} is defined to be the number of Internet routers (including gateways) plus one.

Table 3: The numbers of rollbacks in optimistic distributed cosimulation.

N_{hop}	H.263		JPEG	
	w/o HM	w/ HM	w/o HM	w/ HM
3	7,721	1,775	1,710	380
12	8,697	2,130	1,815	431

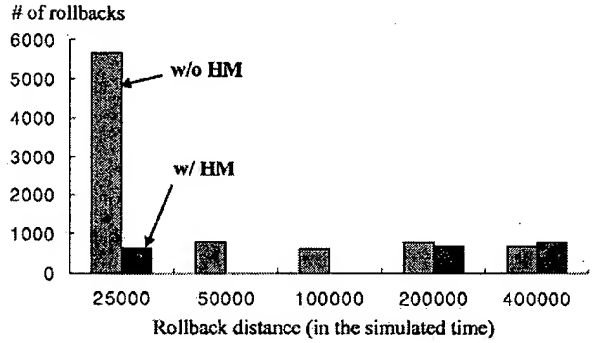


Figure 5: Histograms on rollback statistics (# of rollbacks v.s. rollback distance) in the H.263 decoder example.

construction of HM.

In Table 1 and 2, compared to the run-times of uni-processor **synchronous cosimulation**, the performance improvement of hybrid cosimulation (w/o HM) is mainly from the reduction of HW simulation run-time by using a HW emulator instead of the cycle-based simulator. As shown in Table 4, by applying HM concept to hybrid cosimulation, the numbers of physical messages and rollbacks are reduced down to 0.78% and 4.22% for the H.263 example, respectively (0.84% and 4.87% for the JPEG example). Such reduction of the numbers of physical messages and rollbacks gives 11.28 times (for H.263) and 5.61 times (for JPEG) performance improvement (when $N_{hop} = 3$) in Table 1 and 2. In hybrid cosimulation [1], since the increase of communication overhead does not change rollback behavior, Table 4 gives a single number for each type of cosimulation.

In Table 1 and 2, as the communication overhead represented by N_{hop} increases, the run-time of hybrid cosimulation without HM concept gets increased steeply due to the large numbers of physical messages and rollbacks. while HM concept gives much slower increase of run-time.

In Table 1 and 2, HM concept gives better performance improvement in hybrid cosimulation than in optimistic distributed cosimulation. The reason is as follows. In hybrid cosimulation without HM concept, two simulators synchronize at least at every message transfer as described in Section 2. For the simulator to start, it should wait to receive a message (null message or a message having an event) from other simulator(s). On the contrary, in optimistic distributed cosimulation, simulators do not stop to wait for messages. Thus, the reduction of the number of physical messages in the case of hybrid cosimulation has stronger effect on the reduction of the number of simulator synchronization. i.e. the reduction of simulator synchronization overhead including rollback overhead.

6 Conclusion

In this paper, we present hierarchically grouped message concept to reduce the simulator synchronization overhead in geographically distributed timed cosimulation. We obtained significant performance improvement by applying HM concept to geographically distributed cosimulation environments. Our experiments show that HM concept enables geographically distributed timed cosimulation to be applied in practical situations.

Table 4: The numbers of physical messages (No. MSG) and rollbacks (No. RB) in hybrid cosimulation.

	H.263		JPEG	
	w/o HM	w/ HM	w/o HM	w/ HM
No. MSG	684,262	5,325	163,880	1,375
No. RB	41,091	1,735	10,374	505

Currently, we are integrating hybrid and optimistic distributed cosimulation together with HM concept into an existing system design framework. Our future work includes developing efficient synchronization methods in hybrid distributed cosimulation environments where software. simulators, hardware simulators, and analog simulators co-exist.

Acknowledgement

We would like to thank Prof. Dong S. Ha in the Dept. of ECE, Virginia Tech. for his help with the experimental setup.

References

[1] S. Yoo and K. Choi, "Synchronization Overhead Reduction in Timed Cosimulation", *Proc. IEEE International High Level Design Validation and Test Workshop*, pp. 157-164, Nov. 1997.

[2] K. Hines and G. Borriello, "Selective Focus as a Means of Improving Geographically Distributed Embedded System Co-simulation", *Proc. Eighth IEEE International Workshop on Rapid System Prototyping*, pp. 58–62, June 1997.

[3] K. Hines and G. Borriello, "A Geographically Distributed Framework for Embedded System Design and Validation", *Proc. Design Automat. Conf.*, pp. 140–145, June 1998.

[4] H. Rajaei, R. Ayani, and L. Thorelli, "The Local Time Warp Approach To Parallel Simulation", *Proc. 7th Workshop on Parallel and Distributed Simulation*, pp. 119–126, 1993.

[5] S. Yoo and K. Choi, "Optimistic Distributed Timed Cosimulation Based on Thread Simulation Model", *Proc. Int. Workshop on Hardware-Software Codesign*, pp. 71–75, Mar. 1998.

[6] D. Jaggar, *Advanced RISC Machines Architectural Reference Manual*, Prentice Hall, July 1996.

[7] Telenor, *Telenor's H.263 Software*, http://www.nta.no/brukere/DVC/h263_software/.

[8] Portable Video Research Group, *PVRG-JPEG CODEC*, ftp://havefun.stanford.edu/pub/jpeg/JPEGv1.2.1.tar.Z.

[9] W. Jang, D. Lim, and S. Yoo, *ARM7 Instruction Set Simulator*, http://poppy.snu.ac.kr/Codesign/ARMISS/.

[10] Synopsys, Inc., *Synopsys Simulation Homepage*, http://www.synopsys.com/products/simulation/simulation.html.

[11] D. Lim, K. Rha, and S. Yoo, *Design Automation Lab. Prototyping Board*, http://poppy.snu.ac.kr/Codesign/DAL_P98A/.

A case study on modeling shared memory access effects during performance analysis of HW/SW systems

Source **International Conference on Hardware Software Codesign** archive
Proceedings of the 6th international workshop on Hardware/software codesign table of contents
Seattle, Washington, United States
Pages: 117 - 121
Year of Publication: 1998
ISBN:0-8186-8442-9

Authors Marcello Lajolo
Anand Raghunathan
Sujit Dey
Luciano Lavagno
Alberto Sangiovanni-Vincentelli

Sponsors SIGSOFT: ACM Special Interest Group on Software Engineering
SIGDA: ACM Special Interest Group on Design Automation
IEEE-CS : Computer Society
IFIP : International Federation for Information Processing

Publisher IEEE Computer Society Washington, DC, USA

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 Kei Suzuki , Alberto Sangiovanni-Vincentelli, Efficient software performance estimation methods for hardware/software codesign, Proceedings of the 33rd annual conference on Design automation conference, p.605-610, June 03-07, 1996, Las Vegas, Nevada, United States

2 B. Tabbara and L. Lavagno and A. Sangiovanni-Mncentelli, "Fast Hardware- Software Co-simulation Using Software Synthesis and Estimation," in Proc. Int. High Level Design Validation and Test Wkshp., pp. 149-I 56, Nov. 1997.

3 Subhrajit Bhattacharya , Sujit Dey , Franc Brglez, Performance analysis and optimization of schedules for conditional and loop-intensive specifications, Proceedings of the 31st annual conference on Design automation conference, p.491-496, June 06-10, 1994, San Diego, California, United States

4 Maher Rahmouni , Ahmed A. Jerraya, Formulation and evaluation of scheduling techniques for control flow graphs, Proceedings of European design automation conference with EURO-VHDL '95 on

EURO-DAC '95 with EURO-VHDL '95, p.386-391, September 18-22, 1995, Brighton, England

5 Sujit Dey , Surendra Bommu, Performance analysis of a system of communicating processes, Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design, p.590-597, November 09-13, 1997, San Jose, California, United States

6 Sharad Malik , Margaret Martonosi , Yau-Tsun Steven Li, Static timing analysis of embedded software, Proceedings of the 34th annual conference on Design automation conference, p.147-152, June 09-13, 1997, Anaheim, California, United States

7 R. Ernst , W. Ye, Embedded program timing analysis based on path clustering and architecture classification, Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design, p.598-604, November 09-13, 1997, San Jose, California, United States

8 Kurt Keutzer, Hardware/software co-simulation, Proceedings of the 31st annual conference on Design automation conference, p.439-440, June 06-10, 1994, San Diego, California, United States

9 "Mentor Graphics Seamless CVE Home Page (http://www.mentorg.com/seamlessn.".

10 S. Yoo and K. Choi, "Synchronization Overhead Reduction ini Timed Cosimulation." in Proc. Int. High Level Design Volidationand Test Wkshp., pp. 157-164, Nov. 1997.

11 James A. Rowson , Alberto Sangiovanni-Vincentelli, Interface-based design, Proceedings of the 34th annual conference on Design automation conference, p.178-183, June 09-13, 1997, Anaheim, California, United States

12 Felice Balarin , Massimiliano Chiodo , Paolo Giusto , Harry Hsieh , Attila Jurecska , Luciano Lavagno , Claudio Passerone , Alberto Sangiovanni-Vincentelli , Ellen Sentovich , Kei Suzuki , Bassam Tabbara, Hardware-software co-design of embedded systems: the POLIS approach, Kluwer Academic Publishers, Norwell, MA, 1997

13 J. Buck, S. Ha, E. Lee, and D. Masserchmitt, "Ptolemy: A framework for simulating and prototyping heterogeneous systems," International Joumol on Computer Simulation, SpecialIssue on Simulation Software Management, Jan. 1990.

↑ CITINGS *7*

Kanishka Lahiri , Anand Raghunathan , Sujit Dey, Fast performance analysis of bus-based system-on-chip communication architectures, Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design, p.566-573, November 07-11, 1999, San Jose, California, United States

Abhijit K. Deb , Johnny Öberg , Axel Jantsch, Control and communication performance analysis of embedded DSP systems in the MASIC methodology, Proceedings of the international symposium on Systems synthesis, September 30-October 03, 2001, Montréal, P.Q., Canada

Jong-Yeol Lee , In-Cheol Park, Timed compiled-code simulation of embedded software for performance analysis of SOC design, Proceedings of the 39th conference on Design automation, June 10-14, 2002, New Orleans, Louisiana, USA

Marcello Lajolo, Bus guardians: an effective solution for online detection and correction of faults affecting system-on-chip buses, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, v.9 n.6, p.974-982, 12/1/2001

Sungjoo Yoo , Kyoungseok Rha , Youngchul Cho , Jinyong Jung , Kiyoung Choi, Performance

estimation of multiple-cache IP-based systems: case study of an interdependency problem and application of an extended shared memory model, Proceedings of the eighth international workshop on Hardware/software codesign, p.77-81, May 2000, San Diego, California, United States

Marcello Lajolo , Anand Raghunathan , Sujit Dey, Efficient power co-estimation techniques for system-on-chip design, Proceedings of the conference on Design, automation and test in Europe, p.27-34, March 27-30, 2000, Paris, France

Marcello Lajolo , Anand Raghunathan , Sujit Dey , Luciano Lavagno, Cosimulation-based power estimation for system-on-chip design, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, v.10 n.3, p.253-266, June 2002

↑ INDEX TERMS

Primary Classification:

C. Computer Systems Organization

↳ **C.4** PERFORMANCE OF SYSTEMS

↳ **Subjects:** Performance attributes

Additional Classification:

C. Computer Systems Organization

↳ **C.0** GENERAL

↳ **Subjects:** Hardware/software interfaces

↳ **C.2** COMPUTER-COMMUNICATION NETWORKS

↳ **C.2.2** Network Protocols

↳ **Nouns:** TCP/IP

↳ **C.3** SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS

↳ **Subjects:** Real-time and embedded systems

↳ **C.4** PERFORMANCE OF SYSTEMS

↳ **Subjects:** Modeling techniques

General Terms:

Design, Performance

↑ Collaborative Colleagues:

Sujit Dey:

Jacob Abraham	Michael Cuviello	Luciano Lavagno	Pablo Sanchez
Toshiharu Asaka	Vijay Gangaram	Sharad Malik	Alberto Sangiovanni-Vincentelli
Pranav Ashar	Indradeep Ghosh	Erik Jan Marinissen	Krishna Sekar
Xiaoliang Bai	Rajesh Gupta	Anh Nguyen	Bhaskar Sengupta
Subhrajit Bhattacharya	Zia Iqbal	Debashis Panigrahi	Pradip K. Srimani
Surendra Bommu	Niraj K. Jha	Alice Parker	P. V. Srinivas
Franc Breglez	Faraydon Karim	Miodrag Potkonjak	Clark N. Taylor
Franc Brglez	Gershon Kedem	Anand Raghunathan	Kenneth D. Wagner
Srimat T. Chakradhar	Kamal S. Khouri	Janusz Rajski	Kazutoshi Wakabayashi
Rajit Chandra	Keesup Kim		
Li Chen	Haluk Konuk		
	Angela Krstic		
	Kanishka Lahiri		

	Kwang-Ting Cheng Ying Cheng C.-H. Chia Carla Chiasserini Kwang-Ting Chueng	Wei-Cheng Lai Marcello Lajolo Ganesh Lakshminarayana	Ramesh Rao Srivaths Ravi Michael J. Rodgers Mike Rodgers Steven G. Rothweiler Kaushik Roy Rabindra K. Roy	Weidong Wang Masaaki Yoshida Yi Zhao Yervant Zorian yervant Zorian Yi Z. hao
Marcello Lajolo:	Sujit Dey Luciano Lavagno Mihai Lazarescu Jie Liu Anand Raghunathan Matteo Sonza Reorda Alberto Sangiovanni-Vincentelli Massimo Violante			
Luciano Lavagno:	Alok Agrawal Alex Felice Balarin Jwahar R. Bammi Peter A. Beerel Gérard Berry Bishnupriya Bhattacharya Ivan Blunno Gaetano Borriello Geatano Borriello Robert K. Brayton Jerry Burch Gianpiero Cabodi Paolo Camurati Stefano Cardelli Andre Chatelain Rong Chen Massimilano Chiodo Massimiliano Chiodo Pai Chou Nanette Collins Jordi Cortadella Tullio Cuatto Antonino Damiano Srinivas Devadas Sujit Dey Stephen Edwards Daniel Engels	Enrica Filippi Limor Fix Ian Getreu Paolo Giusto Paolo Guisto Rajesh Gupta Ed Harcourt Edwin Harcourt Youpyo Hong Harry Hsieh Harry C. Hsieh Xiaobo (Sharon) Hu Ahmed A. Jerraya Ahmed Amine Jerraya Attila Jurecska Kurt Keutzer Chunghee Kim Michael Kishinevsky Albert M. Koelmans Alex Kondratyev Wido Kruijtzer Alberto La Rosa M. Lajolo Marcello Lajolo Mihai Lazarescu Mihai T. Lazarescu Edward A. Lee Antonio Lioy	Jean Louis-Guerin Enrico Macii Sharad Malik Radu Marculescu Grant Martin Marc Massot Yves Mathys Patrick C. McGeer Raj S. Mitra Jose Monteiro Sandra Moral Praveen Murthy Amit Nandi Sergio Nocco Steven M. Nowick Ross B. Ortega Claudio Passerone Roberto Passerone Claudio Passeronge Enric Pastor Marco A. Peña Marta Pietkiewicz-Koutny Giovanni Placido Massimo Poncino Stefano Quer Jan Rabaey Anand Raghunathan Matteo Sonza Reorda	Hiroshi Saito Alexander Saldanha Vincentelli Sangiovanni A. Sangiovanni-Vincentelli Alberto Sangiovanni-Vincentelli Alberto L. Sangiovanni-Vincentelli Alberto Sangiovanni-vincentelli Claudio Sansoè Bran Selic Alex Semenov Ellen Sentovich Ellen M. Sentovich Ellen Sentovicha Marco Sgroi Kei Suzuki Bassam Tabbara Alexander Taubin Frank Vahid Massimo Violante Yosinori Watanabe Alex Yakovlev Alexander Yakovlev Alexandre Yakovlev Alexandre V. Yakovlev

				Guang Yang
Anand Raghunathan:	Pranav Ashar Subhrajit Bhattacharya J. Borel Srimat Chakradhar Srimat T. Chakradhar Li Chen Carla Chiasserini Sujit Dey Robert P. Dick Milos D. Ercegovac	Indradeep Ghosh Michael Howells Michael S. Hsiao Chao Huang Niraj K. Jha Kamal S. Khouri Kanishka Lahiri Marcello Lajolo Ganesh Lakshminarayana Luciano Lavagno	Akira Mukaiyama Debashis Panigrahi Nachiketh Potlapally Nachiketh R. Potlapally Vijay Raghunathan Janusz Rajski Ramesh Rao Srivaths Ravi Kaushik Roy Alberto Sangiovanni-Vincentelli	Murugan Sankaradass Jim Sproch Mani B. Srivastava Fei Sun Kenneth D. Wagner Kazutoshi Wakabayashi Weidong Wang
Alberto Sangiovanni-Vincentelli:	D. K. Arvind Adnan Aziz Felice Balarin Massimo Baleani Kaustav Banerjee Martin Baynes Mark Beardslee Gérard Berry Douglas Braun Robert Brayton Robert K. Brayton Jerry Burch Misha Burich Jeffrey Burns Raul Camposano Stefano Cardelli Erik Carlson Giorgio Casinovi Andrea Casotto Henry Chang Edoardo Charbon Rong Chen Massimilano Chiodo Massimiliano Chiodo Claudionor Coelho Ron Collett Nanette Collins Jordi Cortadella Tullio Cuatto Antonino Damiano Luca Daniel Srinivas Davadas Alper Demir Sujit Dey Stephen Edwards	Masahiro Fujita Frank Gennari Ranjit Gharpurey Paolo Giusto Richard Goering Carlo Guardiani Roberto Guerrieri Paolo Guisto William Heller Dave Hightower Harry Hsieh Harry C. Hsieh Chenming Hu Jawahar Jain Yunjian Jiang Attila Jurecska Timothy Kam Masamichi Kawarabayashi Kurt Keutzer Sunil P. Khatri Chunghee Kim Desmond Andrew Kirkpatrick Alex Kondratyev Phil Koopman Marcello Lajolo Jim Lansford Ulrich Lauther Luciano Lavagno Steve Law Mihai Lazarescu Edward Lee Edward A. Lee Bill Lin Jie Liu Hi Keung Ma Hi-Keung Ma Enrico Malavasi Abdul Aziz Malik	Sharad Malik Maq Mannan Radu Marculescu Grant Martin Jonathan Martin Marc Massot Kartikeya Mayaram Patrick C. McGeer Rick McGeer Ken McMillan Amit Mehrotra Robert G. Meyer Robert S. Meyer Trevor Meyerowitz Giovanni De Micheli Paolo Miliozzi Sandra Moral Rajeev Murgai Amit Nandi Amit Narayan Arnit Narayan A. Richard Newton Yoshihito Nishizaki Arlindo L. Oliveira Ralf H. J. M. Otten Davide Pandini Claudio Passerone Roberto Passerone Claudio Passeronge Yatish Patel Claudio Pinello	Richard L. Rudell Jagesh V. Sanghavi A. Sangiovanni-Vincentelli Claudio Sansoè Steve Sapiro Larry Saunders Hamid Savoj Carl Sechen Ellen Sentovich Ellen M. Sentovich Marco Sgroi Henry Sheng Narendra Shenoy Thomas Shiple Thomas Robert Shiple Steven E. Shulz Mário J. Silva Kanwar Jit Singh Vigyan Singhal Kei Suzuki Abdallah Tabbara Bassam Tabbara Johan Van Ginderdeuren Iasson Vassiliou Tiziano Villa Yosinori Watanabe Donald Webber Ruey-Sing Wei Jacob White Jacob K. White Wayne Wolf Hormoz Yaghutiel Alexandre Yakovlev Guang Yang Naeem Zafar

Daniel Engels
Eric Felt
Jerry Fiddler

Jan Rabaey
Anand Raghunathan
Rajeev K. Ranjan
Stephen Ricca
Howard S. Rifkin
Fablo Romeo
James A. Rowson

Stefano Zanella

↑ **Peer to Peer - Readers of this Article have also read:**

- Process migration
  ACM Computing Surveys (CSUR) 32, 3

- Three-dimensional object recognition
  ACM Computing Surveys (CSUR) 17, 1
  Paul J. Besl , Ramesh C. Jain

- Image inpainting
  Proceedings of the 27th annual conference on Computer graphics and interactive techniques
  Marcelo Bertalmio , Guillermo Sapiro , Vincent Caselles , Coloma Ballester

- Three-dimensional medical imaging: algorithms and computer systems
  ACM Computing Surveys (CSUR) 23, 4
  M. R. Stytz , G. Frieder , O. Frieder

A case study on modeling shared memory access effects during performance analysis of HW/SW systems

Marcello Lajolo *
Politecnico di Torino
Torino, Italy
lajolo@polito.it

Anand Raghunathan
NEC C&C Research Labs
Princeton, NJ, USA
anand@ccrl.nj.nec.com

Sujit Dey*
UC San Diego
La Jolla, CA, USA
dey@ece.ucsd.edu

Luciano Lavagno
Politecnico di Torino
Torino, Italy
lavagno@polito.it

Alberto Sangiovanni Vincentelli
University of California at Berkeley
Berkeley, CA, USA
alberto@eecs.berkeley.edu

Abstract

Behavioral simulation with timing annotations derived from performance modeling and analysis is a promising alternative for use in evaluating system-level design tradeoffs [1, 2]. The accuracy of such approaches is determined by how well the effects of various HW and SW architectural features, like the Real Time Operating System (RTOS), shared memories and buses, HW/SW communication mechanisms, etc *are modeled at this level.*
We present a study of the effects of shared memory buses during system-level performance analysis in the POLIS co-design environment, using the example of a TCP/IP Network Interface System. We demonstrate how the effects of the memory arbiter and shared memory bus can be modeled efficiently at the behavioral level, and used to evaluate various design tradeoffs. Experimental results demonstrate that modeling these effects can significantly increase the accuracy of system-level performance estimates.

1 Introduction

Efficient exploration of system-level design tradeoffs depends heavily on the availability of fast and accurate estimation and modeling techniques, for metrics such as performance, power, and cost, to guide various design decisions. Various techniques have been proposed for performance analysis of hardware [3, 4, 5] and software [6, 7]. In this paper, we focus on performance modeling for mixed HW/SW embedded systems. Hardware-software co-simulation [8] remains the most popular approach to performance estimation for such systems. There are several flavors of hardware-software simulation, with varying degrees of efficiency and accuracy. The techniques that involve simulating (RTL) hardware models of the embedded processor(s) along with the models of the hardware components tend to be the most accurate, but are also the slowest. Moreover, detailed hardware models for embedded processors are often not available to system designers. A popular alternative is to use instruction set simulators (ISS) to simulate the software components of the system, and HDL simulators to simulate the hardware components. Instruction set simulators may be cycle and bit-accurate, or may abstract out some architectural details of the target embedded processor such as pipelines and superscalar ordering for efficiency. The efficiency of this approach may still be limited due to the (assembly or binary instruction) level of detail in software simulation, and the communication overhead required to synchronize the execution of the ISS and hardware simulator. While there has been some work on attempting to reduce the synchronization overhead [9, 10], such approaches are still not very efficient for use in exploring tradeoffs during HW/SW co-design. Bus functional models of the embedded processors may be used to exercise the hardware components without needing to run an ISS concurrently, however, only the hardware functionality is simulated in this approach, making it more suitable for validation of the hardware and HW/SW interface. Using an interface-based design methodology [11] helps separate the behavior of the components from their interface protocols, and allows the use of time and space abstractions for efficient validation and analysis.

Behavioral simulation coupled with timing annotations based on performance modeling techniques offers a promising alternative for use in evaluating system-level design tradeoffs [12, 2]. In such approaches, behavioral models of the software components are simulated, and performance estimates for blocks of code are used to annotate timing information. In the POLIS co-design environment [12], a ho-

*This work was started when the authors were at NEC C&C Research Labs, Princeton, NJ

mogeneous behavioral representation is used for hardware as well as software components. The behavioral simulation, analysis, and evaluation is performed using the PTOLEMY heterogeneous simulation environment [13]. Timing information for software modules during simulation is maintained based on performance estimates derived using the technique presented in [1]. The accuracy of behavioral simulation based approaches is determined by how well the effects of various HW and SW architectural features, like the Real Time Operating System (RTOS), shared memories and buses, HW/SW communication mechanisms, *etc* are modeled at this level. For example, the effects of the RTOS are modeled in POLIS during performance analysis, and the user can select between several scheduling policies (*e.g.* round-robin, static priority based, *etc.*) and evaluate their impact on the system performance.

In this paper, we focus on modeling the effects of shared memory buses during system-level performance analysis, using the POLIS co-design environment. The performance of several designs, including graphics and telecommunications applications, may be dominated by memory accesses, making it important to accurately model memory-related effects during system-level design exploration. Using the example of a TCP/IP Network Interface System, we illustrate how the effects of the memory arbiter and shared memory bus can be modeled efficiently at the behavioral level, and used to evaluate various design tradeoffs. Experimental results are presented to indicate that ignoring the effects of the shared memory access bus would have led to significantly incorrect performance estimates, and possibly incorrect design decisions.

The paper is organized as follows. Section 2 provides some background about the TCP/IP Network Interface System used for our study, and the modelling of the system in the POLIS co-design environment. Section 3 presents the results of the evaluation of the effects of the shared memory bus on several design tradeoffs, and section 4 concludes the paper and discusses future work.

2 The TCP/IP System Model

This section provides some background relating to the TCP/IP system, and presents the model used for the system in the POLIS environment.

2.1 Background

A TCP packet consists of three parts:

- An IP header containing, among other fields, the source and destination IP address. The IP header is *usually*, but not always, 20 bytes long,
- A TCP header, containing TCP-specific information. This is usually another 20 bytes,
- The payload, a variable number of bytes (possibly odd) up to a maximum of 65535 bytes.

The TCP/IP protocol requires various tasks to be performed on incoming and outgoing packets, and to maintain the system state. We focus on the evaluation of a dedicated hardware implementation for one of the tasks that is part of the TCP layer - checksum computation. The factors that make this task a good candidate for hardware implementation are explained later.

The IP header is protected by its own 16 bits checksum, that is computed in the IP layer. Since this is computed over such a small number of bytes, it is (relatively) cheap even in software. The TCP data has a 16 bits checksum, carried in the TCP header. It is computed over:

- The 8 bytes of IP address and 16 bits of length field in the IP header,
- The TCP header excluding the 16 bits checksum,
- The payload, taken 16 bits at a time, padding the last byte as NULL if required.

The checksum treats the bytes in pairs, taking each pair of bytes as a 16 bits integer in *big-endian* byte order. Each 16 bits number is added in to the temporary result using unsigned 32 bit integer arithmetic. To obtain the final checksum, the most significant 16 bits of the temporary result are added to the least significant 16 bits, and the result is XOR'ed with $0xffff$.

The checksum computation is particularly inefficient on *little-endian* processors because the *big-endian* 16 bit numbers have to be generated by `shift-or` logic. Also, it is basically a repetitive operation over potentially large volumes of data and contains several bit-level operations. The above factors make the checksum computation a good candidate for hardware implementation. We attempted to model parts of the TCP/IP system relating to the checksum computation using POLIS with the motivations of quantitatively evaluating (i) the performance improvement obtained by implementing the checksum computation in HW, and (ii) the

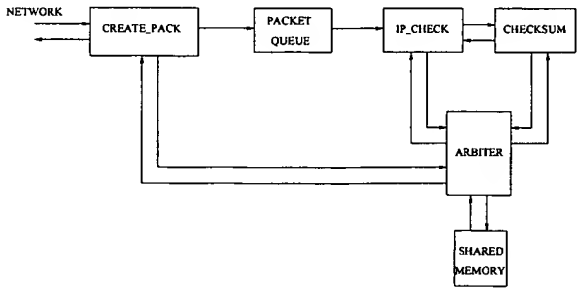


Figure 1. The modeled TCP/IP sub-system

possible adverse effects of SW and HW processes conflicting for accessing the shared packet memory. However, we believe that the effects of shared memory access on system-level performance evaluation that we present are applicable to any HW/SW system, and not limited to the design example or HW/SW configuration used for this study.

2.2 Modeling the TCP/IP subsystem in POLIS

Figure 1 shows the sub-system that has been described in POLIS for our case study. The system was modeled as ten interconnecting CFSMs, each specified in ESTEREL, and their interconnection was described graphically with the Ptolemy user interface.

For incoming packets, the module `create_pack` receives a packet from the lower layer (in this case, the IP layer), and stores it in the shared memory. When it finishes, it sends the information about the starting address of the packet in memory, the number of bytes and the checksum header to a queue (`packet queue`). From this queue, the module `ip_check` retrieves a new packet, overwrites parts of the checksum header (which should not be used in the checksum computation) with 0s, and signals to the `checksum` process that a new packet can be checked for checksum consistency. The `checksum` process performs the core part of the checksum computation, accessing the packet in memory through the arbiter and accumulating the checksum for the packet body. When it is done, it sends the computed 16-bit checksum back to the `ip_check` process, which then compares the computed checksum with the incoming transmitted checksum, and flags an error if they do not match. The flow for outgoing packets is similar, but in the reverse direction, and there is no need for comparison of the final checksum.

2.3 Behavioral Model of the Memory Bus and Arbiter

In the original behavioral description that was used to validate the functionality of the processes, memory accesses were modeled by access to a global array, using a C function call from Esterel, *i.e.* the module `arbiter` shown in Figure 1 was not present. However, as we show in Section 3, using the same model for performance evaluation suffers from the drawback of ignoring effects such as shared memory access conflicts, block access mode (DMA), *etc.* Hence, we described a behavioral model of the shared bus and memory arbiter (shown as module `arbiter` in Figure 1) to model the effect of the controller (arbiter) of the shared memory bus. The `arbiter` module is the only module that can access the shared memory: it receives requests from the processes `create_pack`, `ip_check` and `checksum`, and is responsible for deciding which module is given access to the memory. The functional model of the arbiter is such that the access priority scheme can be easily changed or parametrized. For example, we may specify that in the case of simultaneous requests, the arbiter should give higher priority to `checksum` and lower priority to `create_pack`.

In our system, the primary functions of the arbiter are: (i) to avoid multiple components simultaneously driving the bus in an attempt to access memory using a simple request-grant protocol, (ii) to resolve simultaneous access attempts based on priorities that can be specified by the designer, (iii) to allow components to request dedicated access of the memory bus for a certain number of bus cycles (block access mode or DMA mode). We have created a behavioral model of the arbiter and shared memory bus in Esterel that is called `arbiter` in Figure 1. The `arbiter` process has a dedicated interface to each of the processes that require to access memory, that can be similar to, or an abstraction of, the shared memory bus interface. In addition, each process that accesses memory is enhanced to include an arbiter interface. For example, the signals that interface the `arbiter` process to the `checksum` process are shown in Figure 2. The interface consists of a memory access

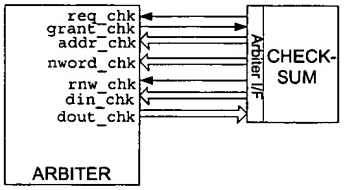


Figure 2. The interface of the arbiter model

request signal *req_chk* on which the `checksum` process generates an event to indicate that it would like to access memory. The starting address is placed on signal *addr_chk*, and a block size signal *nword_chk* is used (in DMA or block access mode) to convey the number of bus cycles of dedicated bus access requested. The arbiter generates and event on the signal *grant_chk* to indicate that the request has been granted. In addition, there are data in, data out, and read/write signals to the memory.

A part of the Esterel specification of the `arbiter` process is shown in Figure 3. Signals *req_create*, *req_ip*, and *req_check* represent the requests for access to the memory bus from the `create_pack`, `ip_check`, and `checksum` processes, respectively. Note that the behavior of the arbiter is described as an infinite loop which immediately encloses a set of nested $if - then - else$ statements that test for the presence of events on the various memory access request signals. The code within this set of $if - then - else$ statements represents the actions to be taken for processing a memory access request from the corresponding module. Figure 3 only shows the code for processing a memory access request from the `checksum` process, the parts for han-

```
....
loop
    if (?req_create=1) then
         ....
         ....  % grant access to create_pack
    elsif (?req_ip=1) then
         ....
         ....  % grant access to create_pack
    elsif (?req_chk=1) then
      i:=?addr_chk;
      emit grant_chk;
      repeat ?nword_chk times
        if (?rnw_chk=false) then
          await din_chk;          % memory write
          emit addr(i); emit din(?din_chk);
          emit rnw(?false);
        else
          emit addr(i);           % memory read
          emit din(?din_chk);
          emit rnw(?true);
          await din_mem;
          emit dout_chk(?din_mem);
        end if;
        i:=i+1;
      end repeat;
      emit res_chk;
    end if;
end loop;
....
```

Figure 3. Esterel model of the `arbiter` process

dling requests from other modules are similar. The priorities given by the arbiter to requests from the various processes are determined by the order in which the request signals are tested in the nested $if - then - else$ statements. For example, the code shown in Figure 3 gives highest priority to requests from `create_pack`, since the signal *req_create* is tested for an event first. Thus, changing the memory access priorities of the processes can be achieved by simply re-ordering the testing of the access request signals in the behavioral arbiter model.

We would like to reiterate that the behavioral arbiter model shown above is not part of the system specification - it was added to model the effects of the shared memory bus and memory arbiter during behavioral level performance simulation. However, during the performance simulation, it is treated just like any other module. The implementation of the `arbiter` process is specified to be HW, because it allows us finer control of its timing properties. The number of memory access cycles, and processing time taken by the arbiter, can be easily modeled using *await tick* statements appropriately inserted in the behavioral model.

3 Performance Simulation and Experimental Results

In the POLIS environment, the system specification, which may consist of a PTOLEMY netlist that describes the interconnection of the functional components or modules and an Esterel specification that describes the functionality of each module, is translated into a network of co-design finite state machines (CFSMs), which are extended FSMs with asynchronous buffered communication. Performance analysis is carried out using the heterogeneous simulation environment offered by PTOLEMY [13]. The performance simulation is based on a C model of each CFSM that is automatically generated, using the hardware/software partitioning specified by the user, the scheduling policy for the RTOS specified by the user, and a timing model for the target processor that is derived during a characterization step [12]. We simulated the TCP/IP subsystem with network traffic that was captured using a profiling tool from an existing software implementation of the TCP/IP protocol.

We performed several experiments to demonstrate the value added by our behavioral model of the arbiter and shared memory bus during system-level design, some of which we present here.

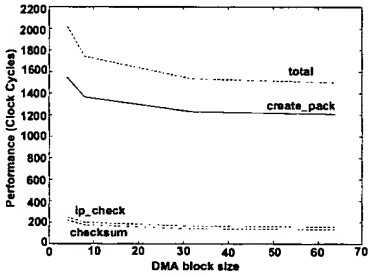


Figure 4. Variation of computation times with DMA block size

In the first experiment we performed an analysis of the variation of the processing times for each module as well as the complete per-packet processing time for the entire system for various sizes of the DMA block size used for memory access. For this experiment, the `create_pack` process was mapped to software running on a MIPS R3000 processor, and the `checksum` and `ip_chk` processes were mapped to hardware. Figure 4 shows the variation of (average) per-packet processing times for the three processes for a test bench consisting of three packets of length 512, 64, and 448 bytes, for block sizes of 4, 8, 32, and 64 bytes. The following conclusions can be drawn from Figure 4:

- As expected, the processing times for all the modules as well as the total processing time decrease with increasing DMA block size, since the handshaking overhead required to obtain memory access is amortized over a

Table 1. Processing times without memory conflicts

packet #	create_pack	ip_check	checksum	total
1	513	1101	1088	2702
2	65	149	136	350
3	449	965	952	2366

larger number of data transfers. The decrease is significant at lower DMA block sizes.

- In addition, the sensitivity of the performance of the software module (`create_pack`) to DMA block size is higher, since the time required for handshaking with the arbiter is much higher for the software module than for the hardware modules.

Note that it would have been impossible to perform the above analysis in the absence of the behavioral model of the shared memory bus and arbiter, since the reported processing times would be constant for various values of block size.

The next experiment we performed was to evaluate the effect of memory conflicts due to the shared memory bus on the performance of the individual processes as well as the overall system performance. The performance estimates without and with memory conflicts are presented in Tables 1 and 2, for a sequence of three packets (512, 64 and 448 bytes long) that are part of a longer stream. The performance estimates without memory conflicts were obtained by not including the arbiter process, and modeling memory as an array shared between the `create_pack`, `ip_check` and `checksum` processes. Access to the shared array is performed using a C function call annotated with a fixed delay to represent the access time of the memory.

Table 2. Processing times with memory conflicts

packet #	create_pack	ip_check	checksum	total
1	513	1617	1538	3688
2	65	218	192	475
3	449	1418	1346	3213

The results indicate that:

- The performance of the `create_pack` process was not affected by the presence of memory conflicts. This is because the memory arbiter gives highest priority to requests from `create_pack` when simultaneous or pending requests are present.

- The per-packet performance estimates of the `ip_check` and `checksum` processes are in error (underestimates) by 46.9% and 41.4%, respectively if memory conflicts are ignored, and the total performance of the system is underestimated by 36.39%

It is clear from the above results that the effects of memory conflicts due to the use of shared memory and the DMA block size need to be considered while estimating the performance of HW/SW systems.

4 Conclusions and Future Work

We presented a case study to study the effects of shared memory buses and arbiters during system-level performance analysis. Using the case study of a part of a TCP/IP network interface system, we have proposed a methodology to model the shared memory bus and arbiter at the behavioral level. We presented experimental results to demonstrate that ignoring these effects leads to a large error in system-level performance estimates, and that the effects of some design tradeoffs cannot be evaluated without modeling memory effects accurately. We are currently working on automatically generating the models required to incorporate the effects of the shared memory bus and memory arbiter during performance analysis of HW/SW systems.

Acknowledgements: The authors would like to thank Leslie French and Toshio Misawa of NEC C&C Research Labs for providing the software implementation of the TCP/IP system, and for useful technical discussions.

References

[1] K. Suzuki and A. Sangiovanni-Vincentelli, "Efficient software performance estimation methods for hardware/software codesign," in *Proc. Design Automation Conf.*, pp. 605–610, June 1996.

[2] B. Tabbara and L. Lavagno and A. Sangiovanni-Vincentelli, "Fast Hardware-Software Co-simulation Using Software Synthesis and Estimation," in *Proc. Int. High Level Design Validation and Test Wkshp.*, pp. 149–156, Nov. 1997.

[3] S. Bhattacharya, S. Dey, and F. Brglez, "Performance analysis and optimization of schedules for conditional and loop-intensive specifications," in *Proc. Design Automation Conf.*, pp. 491–496, June 1994.

[4] M. Rahmouni and A. Jerraya, "Formulation and evaluation of scheduling techniques for control flow graphs," in *Proc. European Design Automation Conf.*, Sept. 1995.

[5] S. Dey and S. Bommu, "Performance analysis of a system of communication processes," in *Proc. Int. Conf. Computer-Aided Design*, pp. 590–597, Nov. 1997.

[6] S. Malik, M. Martonosi, and Y. T. S. Li, "Static Timing Analysis of Embedded Software," in *Proc. Design Automation Conf.*, pp. 147–152, June 1997.

[7] R. Ernst and W. Ye, "Embedded program timing analysis based on path clustering and architecture classification," in *Proc. Int. Conf. Computer-Aided Design*, pp. 598–604, Nov. 1997.

[8] J. Rowson, "Hardware/Software Co-Simulation," in *Proc. Design Automation Conf.*, pp. 439–440, June 1994.

[9] "Mentor Graphics Seamless CVE Home Page (http://www.mentorg.com/seamless/).".

[10] S. Yoo and K. Choi, "Synchronization Overhead Reduction ini Timed Cosimulation," in *Proc. Int. High Level Design Validation and Test Wkshp.*, pp. 157–164, Nov. 1997.

[11] J. Rowson, "Interface Based Design," in *Proc. Design Automation Conf.*, pp. 178–183, June 1997.

[12] F. Balarin, M. Chiodo, H. Hsieh, A. Jureska, L. Lavagno, C.Passerone, A. Sangiovanni-Vincentelli, E. Sentovich, K. Suzuki, and B. Tabbara, *Hardware-software Co-Design of Embedded Systems: The POLIS Approach*. Kluwer Academic Publishers, Norwell, MA., 1997.

[13] J. Buck, S. Ha, E. Lee, and D. Masserchmitt, "Ptolemy: A framework for simulating and prototyping heterogeneous systems," *International Journal on Computer Simulation, Special Issue on Simulation Software Management*, Jan. 1990.

A statechart based HW/SW codesign system

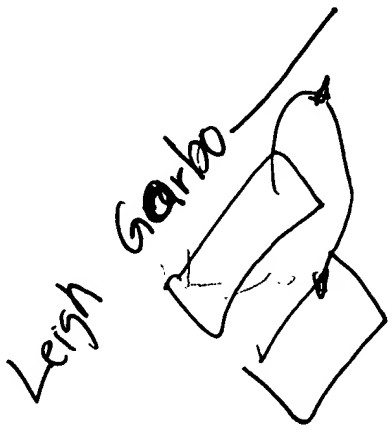
Full text Pdf (385 KB)

Source **International Conference on Hardware Software Codesign** archive
Proceedings of the seventh international workshop on Hardware/software codesign table of contents
Rome, Italy
Pages: 162 - 166
Year of Publication: 1999
ISBN:1-58113-132-1

Authors I. D. Bates Univ. of Newcastle, Newcastle-upon-Tyne, UK
E. G. Chester Univ. of Newcastle, Newcastle-upon-Tyne, UK
D. J. Kinniment Univ. of Newcastle, Newcastle-upon-Tyne, UK

Sponsors IEEE-CS : Computer Society
IFIP : International Federation for Information Processing
SIGSOFT: ACM Special Interest Group on Software Engineering
SIGDA: ACM Special Interest Group on Design Automation

Publisher ACM Press New York, NY, USA



↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 Chiodo M, Giusto P, Jurescska A, Hsieh H C, Sangiovanni-Vincentelli A, Lavagno L., Hardware-Software Codesign of Embedded Systems, August 1994, IEEE Micro.

2 David Harel, Statecharts: A visual formalism for complex systems, Science of Computer Programming, v.8 n.3, p.231-274, June 1, 1987

3 D. Druinsky, D. Harel: Using Statecharts for Hardware Description and Synthesis. IEEE Trans. on CAD, vol. 8, pp798-807, July 1989.

4 K. Buchenrieder et al: Mapping Statechart Models onto an FPGA-Based ASIP Architecture, Proceedings of - EURO-CAD 96, pp 184.189, ISBNiISSN: 08 1867573X

5 Gérard Berry , Georges Gonthier, The ESTEREL synchronous programming language: design, semantics, implementation, Science of Computer Programming, v.19 n.2, p.87-152, Nov. 1992

6 Felice Balarin , Massimiliano Chiodo , Paolo Giusto , Harry Hsieh , Attila Jurecska , Luciano Lavagno , Claudio Passerone , Alberto Sangiovanni-Vincentelli , Ellen Sentovich , Kei Suzuki , Bassam Tabbara, Hardware-software co-design of embedded systems: the POLIS approach, Kluwer Academic Publishers, Norwell, MA, 1997

↑ INDEX TERMS

Primary Classification:
B. Hardware
↳ **B.4** INPUT/OUTPUT AND DATA COMMUNICATIONS

Additional Classification:
F. Theory of Computation
↳ **F.1** COMPUTATION BY ABSTRACT DEVICES

General Terms:
Design, Theory

Keywords:
CFSMs, POLIS, statecharts

↑ Collaborative Colleagues:

I. D. Bates:	E. G. Chester	
	D. J. Kinniment	
E. G. Chester:	I. D. Bates	
	D. J. Kinniment	
D. J. Kinniment:	I. D. Bates	O. V. Maevsky
	F. P. Burns	G. Russell
	T. J. Butler	A. Semenov
	A. Bystrov	D. Shang
	E. G. Chester	F. Xia
	J. N. Coleman	A. Yakovlev
	B. Gao	A. V. Yakovlev
	A. Koelmans	
	A. M. Koelmans	
	Albert Koelmans	

↑ Peer to Peer - Readers of this Article have also read:

- Data structures for quadtree approximation and compression
  Communications of the ACM 28, 9
  Hanan Samet

- A hierarchical single-key-lock access control using the Chinese remainder theorem
  Proceedings of the 1992 ACM/SIGAPP Symposium on Applied computing
  Kim S. Lee , Huizhu Lu , D. D. Fisher

- 3D representations for software visualization
  Proceedings of the 2003 ACM symposium on Software visualization
  Andrian Marcus , Louis Feng , Jonathan I. Maletic

- Probabilistic surfaces: point based primitives to show surface uncertainty
  Proceedings of the conference on Visualization '02
  Gevorg Grigoryan , Penny Rheingans

- Efficient simplification of point-sampled surfaces
  Proceedings of the conference on Visualization '02
  Mark Pauly , Markus Gross , Leif P. Kobbelt

A STATECHART BASED HW/SW CODESIGN SYSTEM

I.D. Bates
EPSRC Engineering Design Centre
University of Newcastle
Newcastle-upon-Tyne NE1 7RU
UK
Tel: +44 191 222 8173
i.d.bates@ncl.ac.uk

E.G. Chester
Dept. of Elect. and Electronic Eng.
University of Newcastle
Newcastle-upon-Tyne NE1 7RU
UK
Tel: +44 191 222 7325
graeme.chester@ncl.ac.uk

D.J. Kinniment
Dept. of Elect. and Electronic Eng.
University of Newcastle
Newcastle-upon-Tyne NE1 7RU
UK
Tel: +44 191 222 7338
david.kinniment@ncl.ac.uk

1. ABSTRACT

The Codesign Finite State Machine [1] (CFSM) formal model provides a suitable approach for the description of hardware/software systems. The POLIS tool from Berkeley implements the CFSM methodology but currently relies on the textually based Esterel specification language as a high level for the description of individual CFSMs. The designer must then use the Ptolemy simulator to interconnect the CFSM network and perform co-simulation. This paper describes work in progress in developing a system which instead aims to use StatemateTM, a statechart based tool for seamless specification and co-simulation of the entire CFSM network, whilst using the POLIS tool for 'C', VHDL code generation and performance estimation. This technique should give the clear advantages of using a graphical specification language together with a uniform co-simulation framework.

1.1 Keywords

Statecharts, POLIS, CFSMs

2. INTRODUCTION

Statecharts[2] are a proven and industrially desirable specification methodology. They should also be suitable for use as a specification language for a hardware/software codesign system. CFSMs use a locally synchronous, globally asynchronous paradigm that has advantages for partitioning a system. Our work has investigated how individual Statechart hierarchies can provide a synchronous model which may represent individual CFSMs in a design. Statechart represented CFSMs may then be connected in a suitable network to form a suitable system model. The Statemate tool may then be used to represent the globally asynchronous paradigm allowing the CFSM network to be simulated.

Several examples of the use of statecharts for automatic hardware and software generation exist. The work of Druinsky and Harel[3] developed techniques for mapping Statechart trees onto programmable devices, and has been developed and used by Buchenrieder[4] since. Several commercially available systems (Statemate and SpecChart code generators) exist, but from a hardware generation aspect require the use of behavioural VHDL as an intermediate step. In his paper[4] Buchenrieder states: 'The Statemate^{TM} code generator produces behavioural VHDL which is often not synthesisable'. The author has verified this whilst attempting to synthesise seemingly trivial Statechart models to hardware via Statemate generated VHDL, using the Cadence DFWII^{TM} software suite.

From a software code generation aspect, to the authors knowledge, none of the existing Statechart based methods and tools allow for code performance estimation on a target processor. This is a major drawback if the tools are to be used as the basis for a codesign system. Similarly from a hardware performance estimation point of view (with the exception of Buchenreiders approach) hardware performance can only be estimated via VHDL.

The Codesign Finite State Machine approach to hardware/software codesign has several benefits. The model is globally asynchronous and this greatly simplifies partitioning of the model into hardware and software modules. Furthermore due to the relatively low level structure of individual CFSMs synthesis of hardware or software is relatively straightforward. The CFSM software synthesis uses the S-Graph as an intermediate step to estimate the performance on a target processor. This process has the advantage of producing portable and efficient code.

However since the CFSM model is at too low a level to be used directly by designers, higher level specification languages are necessary. Suitable languages are the synchronous class of languages which include Esterel[5], statecharts or a subset of VHDL.

The POLIS [6] system developed at the University of California, Berkeley, implements a HW/SW codesign system using the CFSM model as its basis. As the input to the system the intermediate level SHIFT language is used. SHIFT is capable of describing a hierarchical network of Codesign Finite State Machines. It allows for the description of individual CFSMs as reactive finite state machines. Individual CFSMs may then be embedded in a net-list which may reference other CFSMs or arithmetic, boolean or user-defined functions.

The chosen specification language for the POLIS project is Esterel[5]. To complement POLIS an strl2shift converter

CODES '99 Rome Italy

program has been written, and in this way CFSMs may be developed in Esterel, converted into SHIFT and input to the POLIS program for synthesis. Whilst Esterel is a sound synchronous language with a good debugger, it does not offer the design advantages of a good graphical language such as statecharts. There is also a further drawback as having first entered an Esterel description of each CFSM the designer must next use the separate Ptolemy tool to interconnect CFSMs and perform co-simulation.

3. USING STATECHARTS TO MODEL COMMUNICATING CFSM NETWORKS

Using our methodology each individual CFSM in a network of interconnected CFSMs is modelled by a separate statechart hierarchy. The interconnections between statecharts (in the form of propagated events and signals) may be specified using the Statemate statechart package's Activity Charts as illustrated in Figure 1. Flows between statecharts are used to represent events and signals. In this example of a paint drying system the Statecharts PAINT_SYS_CTRL and BLOWER_SYS are referenced as providing the behaviour for each CFSM.

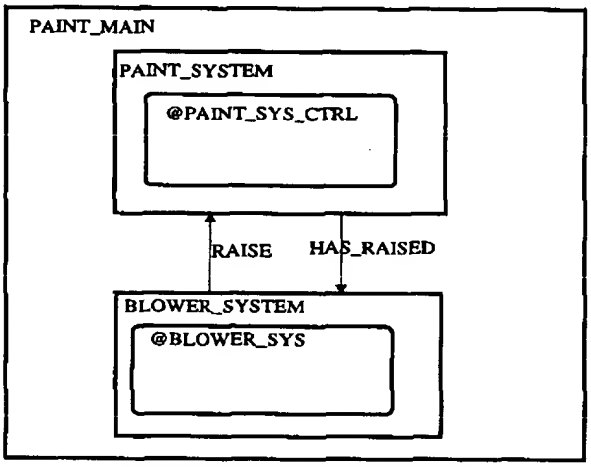


Figure 1. Statemate Activity Chart

3.1 Modelling Asynchronous CFSMs in Statemate

Statemate uses Activity Charts to allow the modelling of complex systems composed of many activities, each of which contain a behavioural specification in the form of a Statechart. The interaction of multiple Statechart hierarchies is controllable via start, stop, suspend and resume commands which may be issued by controlling Statecharts. The CFSM network must eventually be implemented in a combination of software and hardware. In the case of software a scheduling policy must be implemented on the target system. We have therefore modelled a basic scheduler in Statemate, allowing control of the interaction of events between various CFSMs.

The scheduler consists of a Statechart (Figure 2) which implements the functionality of a scheduler and controls the buffering of system inputs and outputs. Every simulated clock cycle, the scheduler triggers a Statemate Activity which detects any system input changes and places them in an input buffer modelled in Statemates language using an array. This activity is continuously executed THROUGH-OUT the system being in state IDLE. After each execution of this activity the Statechart monitors the input arrays for changes in input signal corresponding to each CFSM. If any changes are detected then the scheduler algorithm is run. This scheduler algorithm is specified by the statechart DO_SCHEDULE.

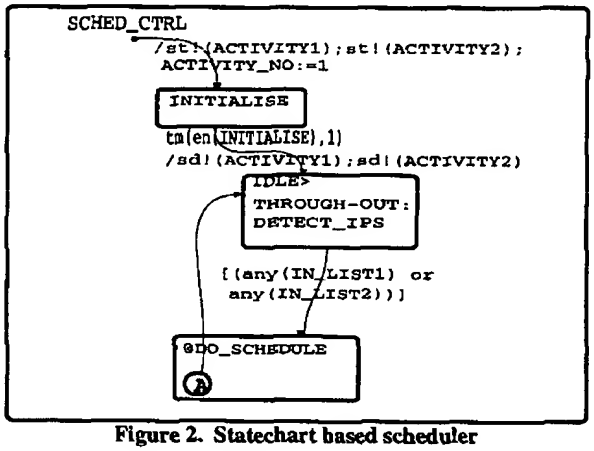


Figure 2. Statechart based scheduler

The simple algorithm presented here in Figure 3 allows for scheduling of just two activities but could be expanded for more or possibly 'n' activities.

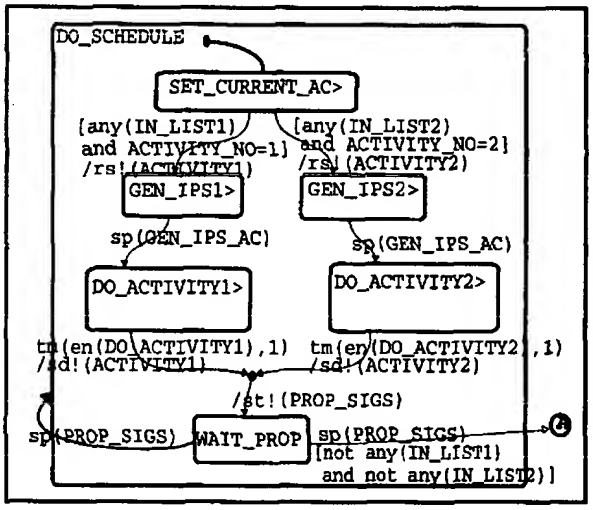


Figure 3. Statechart for scheduler algorithm

A static reaction (code triggered when entering a state) within state SET_CURRENT_AC chooses the next activity to be scheduled in a 'round robin' fashion. Two transitions then use the Statemate rs!(A) 'resume activity' command to execute the given task if it currently has pending input event(s).

The GEN_IPS state triggers a Statemate activity which propagates pending input events to the relevant activity being scheduled. The system remains in this state until all events/signals have been propagated. The system then enters the DO_ACTIVITYn state wherein ACTIVITYn is executed for a simulation clock cycle. The Statemate toolset supports two 'time models' and in this case if its Asynchronous time model option is selected, one clock cycle corresponds to as many simulation 'micro-steps' so as to result in a new stable system stable state. So resuming execution for one cycle results in a single transition in the selected activities statechart. Whilst the chosen activity executes another activity DETECT_OPS is enabled to detect output changes in the selected statechart and store them in an output signal buffer. Once the chosen activity has executed for a simulation cycle the activity is suspended. The activity PROP_SIGS is next executed which effectively propagates any pending active outputs in the output buffer back to the relevant input buffer (so that signals may be exchanged between the scheduled tasks). If any active pending signals are then found in the input buffer the DO_SCHEDULE state is re-entered and the scheduling algorithm is repeated, otherwise the state is exited and the system enters the IDLE state detecting any further system inputs.

3.2 Embedding performance data in the statechart model

The above method allows statecharts to be used to simulate a CFSM network. At present by using tools we have developed it is possible to generate a CFSM description in the Berkeley SHIFT format from the Statechart description of the CFSM network, and use the POLIS system to perform performance estimation and synthesize software and hardware. In future work we aim to embed performance data back into the Statechart model for use in co-simulation.

4. GENERATION OF A SHIFT SPECIFICATION FROM A STATECHART MODEL

In order to use our methodology of representing a CFSM model using Statecharts in a codesign system, we interfaced it to the POLIS package to facilitate performance estimation and hardware/software synthesis. Since the POLIS system supports Esterel we initially considered generating Esterel code from our statechart model.

4.1 Generating Esterel from a statechart model

Representing a statechart design using Esterel is not as straightforward as might be thought. This is mainly due to the fact that a statechart transition may be triggered by either event occurrence, a condition variables value or a combination of these two. This in effect means that any state in a Statechart having transitions labelled with more than just basic events, must be represented in an Esterel program using a loop which, continuously checks the status of the condition variable. Furthermore in order to devise a methodology suitable for automated code generation it was necessary to explicitly encode statechart state variables in the Esterel code. This is because statecharts allow transitions directly into sub-states of a state and visa versa, and this is difficult to represent naturally in an equivalent Esterel program.

Taking these considerations we devised a suitable methodology for the generation of Esterel from statecharts. Unfortunately when this was applied to simple examples, although the Esterel code could accurately represent a statechart model, the resulting SHIFT description (obtained using the POLIS strl2shift utility) was unreasonably large. This results in poor performance when hardware or software is synthesized. For this reason using Esterel code as an intermediate step has been discounted.

4.2 Direct generation of SHIFT description from a statechart model

The statechart language provides a large number of different functions and operators that may be used for transition expressions. It is our aim to initially provide support for a reasonably large subset of these in our statechart to SHIFT converter. The system we have developed initially does not handle AND states although we have some ideas of how these could be added in the future. This is not as much a drawback as might be thought however since similar concurrent behaviour can be obtained using multiple Statecharts in an asynchronous manner.

Our converter can handle most common statechart EVENT/ CONDITION expressions, including comparisons involving data items.

4.2.1 Handling Hierachy

When generating the SHIFT representation we flatten each statechart into an FSM representable in the SHIFT language.

Hierarchical transitions are probably the major advantage of the statechart language. Our system considers any transition from a non-basic state as implicitly representing a transition from each of the source state's child states. The target state for all transitions must also be represented in the FSM as a basic state so if the transition arrives at a non-basic state that state's default transition is followed. The process of recursive descent is followed until a basic state is found.

4.2.2 Handling transition expressions

Statecharts generally include relatively complicated transition expressions. The use of pure valued or conditional expressions without any event information is also quite common. The CFSM methodology on the other hand assumes that all pure valued signals carry an associated event presence signal. This normally gives an advantage as the CFSM can wait for an event related to a change of valued signal or data item before reacting.

There are however instances where the CFSM must 'self trigger' to be able to correctly handle pure conditional expressions. The Statechart in Figure 4 illustrates such an

example.

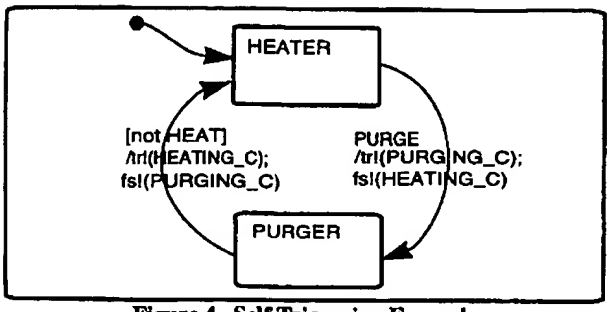


Figure 4. Self Triggering Example

In this example when implementing this Statechart in a CFSM when the CFSM transitions from HEATER to PURGE it must emit a self trigger event to subsequently re-trigger itself. This is necessary due to a need to check if the HEAT value is already false, which would mean that the CFSM must transition immediately back to the HEATER state.

Two commonly used functions in Statecharts are the tr(C) and fs(C) which sense when a condition became true or false. In order to provide such functionality in the CFSM it is necessary to provide double buffering of condition variables which are used in these functions. The CFSM thus generates the 'previous value' of each such input on each transition. These outputs may then be fed back to the CFSM and used at the next time-step. Additional transitions must also be provided so that if the CFSM is idling in a given state then these inputs are still double buffered (even if there is no resulting transition to another state).

To greatly simplify the evaluation of Statechart expression we have used the following rules when generating the CFSM:

- Atomic transitions (those whose expressions are a single event or condition) are handled by feeding the relevant input directly to the relevant CFSM input.
- All remaining transitions are assigned their own separate pure boolean CFSM input and the transition expression is evaluated using a series of combinational functions in the CFSM net, and fed to this input.

4.2.3 Resolving transition priority

The CFSM must correctly resolve the priority of statechart transitions. This is simply achieved because each transition has one and only one input that is used as a trigger. Hence it is easy to resolve transition priorities such that low level transitions can only occur when higher level transition trigger functions are not currently active.

In the case where two transitions existing with the same priority are non-deterministic our converter currently ensures deterministic behaviour in the resulting CFSM by making the two transitions mutually exclusive.

5. USING THE SYSTEM

To date we have completed the described Statechart-CFSM converter and tested it with simple examples. Synthesis of software is satisfactory and we are able to run the POLIS generated code on a UNIX™ workstation.

The size of the resulting SW/HW (from the POLIS software) via our system compares very favourably with that from Esterel code and the appropriate converter. This is proven by a comparison (see Table 1) where we have hand-coded an equivalent description of a statechart in Esterel. Generally our system achieves more efficient results than the Esterel specification when specifications have many non event expressions within them.

Specification method	POLIS SW synth. time (SPARC 20)	SW Costs for 68hc11 (see key)	HW Costs[a] (see key)
Esterel	800 sec	size: 3911 min t: 338 max t: 1128	POLIS failed to synth. within 4hrs
Statechart	8 sec.	size: 843 min t: 223 max t: 511	pi: 41 po: 6 lat: 10 sop: 353 fac: 235
Key for synthesis costs	*size = s/w size in bytes* *min t = min cycle time in seconds* *max t = max cycle time in seconds* *pi/po = no. primary inputs/outputs* *lat = number of latches* *sop/fac = no. literals in sum-of-product or factored form*		

Table 1: Comparison of HW/SW synthesis

a.Technology independant form.

Synthesis of hardware using the POLIS system currently raises some problems. Due to the relatively high ARITHMETIC complexity of such CFSMs we are experiencing unreasonably long hardware synthesis times. The reason for this appears to be that the POLIS system is attempting to flatten all parts of the resulting network and then apply boolean minimisation.

One possible solution to this problem may be to use POLIS generated behavioural VHDL together with commercial synthesis tools. The improved generation of VHDL from the CFSM network is a feature of the new release of the POLIS software.

6. CONCLUSIONS

Statecharts are a convenient specification methodology for

use in conjunction with CFSM theory for the description of individual CFSMs. As they are graphical they are easier for the designer to visualise, and using tools such as Statemate offer powerful simulation capabilities.

Our methodology promises several advantages over the standard use of POLIS with the Esterel language for specification and Ptolemy for co-simulation as a codesign system:

- For the statechart models that we have considered, the resulting the CFSMs are smaller than those obtained standard via Esterel with the POLIS system. In practice this means greater efficiency in S/W and H/W implementations. This is demonstrated by the metrics in Table 1.
- Statecharts are in our view a more industrially desirable specification methodology than Esterel, being graphical in nature.
- The SHIFT file generated from a statechart is close to the original statechart specification. Therefore it may be possible for POLIS generated performance data to be embedded in the original Statemate model in order to provide high-level co-simulation in a uniform environment.
- If the specification and simulation tasks can be integrated a uniform codesign system will be obtained.

It is quite early in our project and at present we only give an outline of the system we aim to produce. To date software to generate the SHIFT code from a Statechart model has been developed. Future work will further develop the use of the Statemate environment for high level co-simulation including performance estimation. We also intend to expand our Statemate based scheduler to allow the simulation of hardware as well as software CFSMs.

Our codesign system will then be proved with a suitable real-world industrial case study.

7. ACKNOWLEDGEMENTS

The authors' would like to thank support given from the EPSRC Engineering Design Centre, University of Newcastle upon Tyne during undertaking this work. Also thanks to Luciano Lavagno, Bassam Tabbara and the POLIS support team for their advice.

8. REFERENCES

[1] Chiodo M, Giusto P, Jurescska A, Hsieh H C, Sangiovanni-Vincentelli A, Lavagno L., Hardware-Software Codesign of Embedded Systems, August 1994, IEEE Micro.

[2] D. Harel: Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming, vol 8, pp 231-274, 1987.

[3] D. Druinsky, D. Harel: Using Statecharts for Hardware Description and Synthesis. IEEE Trans. on CAD, vol. 8, pp798-807, July 1989.

[4] K. Buchenrieder et al: Mapping Statechart Models onto an FPGA-Based ASIP Architecture, Proceedings of EURO-CAD 96, pp 184-189, ISBN/ISSN: 081867573X

[5] G. Berry et al: The Esterel synchronous programming language: Design, semantics, implementation. Science of Computer Programming, vol. 19(2), pp87-152, 1992.

[6] F. Balarin et al: Hardware-Software Co-design of Embedded Systems - The POLIS approach. Kluwer Academic Publishers, 1997.

Software timing analysis using HW/SW cosimulation and instruction set simulator

Full text Publisher Site Pdf (38 KB)

Source **International Conference on Hardware Software Codesign** archive
Proceedings of the 6th international workshop on Hardware/software codesign table of contents
Seattle, Washington, United States
Pages: 65 - 69
Year of Publication: 1998
ISBN:0-8186-8442-9

Authors Jie Liu
Marcello Lajolo
Alberto Sangiovanni-Vincentelli

Sponsors SIGSOFT: ACM Special Interest Group on Software Engineering
SIGDA: ACM Special Interest Group on Design Automation
IEEE-CS : Computer Society
IFIP : International Federation for Information Processing

Publisher IEEE Computer Society Washington, DC, USA

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 Felice Balarin , Massimiliano Chiodo , Paolo Giusto , Harry Hsieh , Attila Jurecska , Luciano Lavagno , Claudio Passerone , Alberto Sangiovanni-Vincentelli , Ellen Sentovich , Kei Suzuki , Bassam Tabbara, Hardware-software co-design of embedded systems: the POLIS approach, Kluwer Academic Publishers, Norwell, MA, 1997

2 Ptolemy Home Page, "http://ptolemy.eecs.berkeley.edu"

3 Esterel Home Page, "http://www.inria.tr/meije/esterel"

4 R. Ernst , W. Ye, Embedded program timing analysis based on path clustering and architecture classification, Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design, p.598-604, November 09-13, 1997, San Jose, California, United States

5 K. ten Hagen, H. Meyr, "Timed and Untimed Hardware Software Cosimulation: Application and Efficient Implementation", Proceedings of Int. Workshop on Hardware-Software Codesign, Oct. 1993

6 E. A. Lee and A. Sangiovanni-Vincentelli, "A Denotational Framework for Comparing Models of

Computation", ERL Memorandum UCB/ERL M97/11, University of California, Berkeley, CA 94720, January 30, 1997.

7 S. Leef, "Hardware and Software Co-Verification - Key to Co- Design", Electronic Design, September 15,97, PP 67-7 1

8 C. Passerone, L.Lavagno, etc. "Trade-off Evaluation in Embedded System Design via Co-simulation". Proceedings of ASP-DAC, PP 291-297,1997.

9 Kurt Keutzer, Hardware/software co-simulation, Proceedings of the 31st annual conference on Design automation conference, p.439-440, June 06-10, 1994, San Diego, California, United States

10 Kei Suzuki , Alberto Sangiovanni-Vincentelli, Efficient software performance estimation methods for hardware/software codesign, Proceedings of the 33rd annual conference on Design automation conference, p.605-610, June 03-07, 1996, Las Vegas, Nevada, United States

11 "ST20 'Osprey' Toolset: User Manual", SGS-THOMSON Electronics, March 1997

12 S. Yoo, K. Choi, "Synchronization Overhead Reduction in Timed Cosimulation" Proceedings of Int. High Level Design Validation.

↑ CITINGS *11*

Marcello Lajolo , Mihai Lazarescu , Alberto Sangiovanni-Vincentelli, A compilation-based software estimation scheme for hardware/software co-simulation, Proceedings of the seventh international workshop on Hardware/software codesign, p.85-89, March 1999, Rome, Italy

M. Meerwein , C. Baumgartner , T. Wieja , W. Glauert, Embedded systems verification with FGPA-enhanced in-circuit emulator, Proceedings of the 13th conference on International Symposium on System Synthesis, September 20-22, 2000, Madrid, Spain

R. Ruelland , G. Gateau , T. Meynard , J. C. Hapiot, Digital emulator and observer of multicell converter, Mathematics and Computers in Simulation, v.63 n.3-5, p.335-347, 17 November 2003

Per Bjuréus , Axel Jantsch, Performance analysis with confidence intervals for embedded software processes, Proceedings of the international symposium on Systems synthesis, September 30-October 03, 2001, Montréal, P.Q., Canada

Benoit Clement , Richard Hersemeule , Etienne Lantreibecq , Bernard Ramanadin , Pierre Coulomb , Francois Pogodalla, Fast prototyping: a system design flow applied to a complex system-on-chip multiprocessor design, Proceedings of the 36th ACM/IEEE conference on Design automation conference, p.420-424, June 21-25, 1999, New Orleans, Louisiana, United States

J. Jung , S. Yoo , K. Choi, Performance improvement of multi-processor systems cosimulation based on SW analysis, Proceedings of the DATE 2001 on Design, automation and test in Europe, p.749-753, March 2001, Munich, Germany

Francois Pogodalla , Richard Hersemeule , Pierre Coulomb, Fast prototyping: a system design flow for fast design, prototyping and efficient IP reuse, Proceedings of the seventh international workshop on Hardware/software codesign, p.69-73, March 1999, Rome, Italy

Jeffry T. Russell , Margarida F. Jacome, Architecture-level performance evaluation of component-based embedded systems, Proceedings of the 40th conference on Design automation, June 02-06, 2003, Anaheim, CA, USA

G. Beltrame , C. Brandolese , W. Fornaciari , F. Salice , D. Sciuto , V. Trianni, Modeling assembly instruction timing in superscalar architectures, Proceedings of the 15th international symposium on System Synthesis, October 02-04, 2002, Kyoto, Japan

Marcello Lajolo , Anand Raghunathan , Sujit Dey, Efficient power co-estimation techniques for system-on-chip design, Proceedings of the conference on Design, automation and test in Europe, p.27-34, March 27-30, 2000, Paris, France

Marcello Lajolo , Anand Raghunathan , Sujit Dey , Luciano Lavagno, Cosimulation-based power estimation for system-on-chip design, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, v.10 n.3, p.253-266, June 2002

↑ INDEX TERMS

Primary Classification:

C. Computer Systems Organization

↳ **C.0** GENERAL

↳ **Subjects:** Instruction set design (e.g., RISC, CISC, VLIW)

Additional Classification:

C. Computer Systems Organization

↳ **C.0** GENERAL

↳ **Subjects:** Hardware/software interfaces

↳ **C.3** SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS

↳ **Subjects:** Real-time and embedded systems

↳ **C.4** PERFORMANCE OF SYSTEMS

↳ **Subjects:** Performance attributes

General Terms:

Algorithms, Design, Measurement, Performance

↑ Collaborative Colleagues:

Marcello Lajolo:	Sujit Dey	
	Luciano Lavagno	
	Mihai Lazarescu	
	Jie Liu	
	Anand Raghunathan	
	Matteo Sonza Reorda	
	Alberto Sangiovanni-Vincentelli	
	Massimo Violante	
Jie Liu:	Bob Broeg	David Olson
	Patrick Cheung	Vikram A. Saletore
	R. Currey	Alberto Sangiovanni-Vincentelli
	Leonidas Guibas	
	Marcello Lajolo	
	Yiu B. Lam	Feng Zhao

Edward A. Lee
Ted G. Lewis
Theodore G. Lewis
John Marsaglia

Hai Zhuge

Alberto Sangiovanni-Vincentelli:

D. K. Arvind
Adnan Aziz
Felice Balarin
Massimo Baleani
Kaustav Banerjee
Martin Baynes
Mark Beardslee
Gérard Berry
Douglas Braun
Robert Brayton
Robert K. Brayton
Jerry Burch
Misha Burich
Jeffrey Burns
Raul Camposano
Stefano Cardelli
Erik Carlson
Giorgio Casinovi
Andrea Casotto
Henry Chang
Edoardo Charbon
Rong Chen
Massimilano Chiodo
Massimiliano Chiodo
Claudionor Coelho
Ron Collett
Nanette Collins
Jordi Cortadella
Tullio Cuatto
Antonino Damiano
Luca Daniel
Srinivas Davadas
Alper Demir
Sujit Dey
Stephen Edwards
Daniel Engels
Eric Felt
Jerry Fiddler

Masahiro Fujita
Frank Gennari
Ranjit Gharpurey
Paolo Giusto
Richard Goering
Carlo Guardiani
Roberto Guerrieri
Paolo Guisto
William Heller
Dave Hightower
Harry Hsieh
Harry C. Hsieh
Chenming Hu
Jawahar Jain
Yunjian Jiang
Attila Jurecska
Timothy Kam
Masamichi Kawarabayashi
Kurt Keutzer
Sunil P. Khatri
Chunghee Kim
Desmond Andrew Kirkpatrick
Alex Kondratyev
Phil Koopman
Marcello Lajolo
Jim Lansford
Ulrich Lauther
Luciano Lavagno
Steve Law
Mihai Lazarescu
Edward Lee
Edward A. Lee
Bill Lin
Jie Liu
Hi Keung Ma
Hi-Keung Ma
Enrico Malavasi
Abdul Aziz Malik

Sharad Malik
Maq Mannan
Radu Marculescu
Grant Martin
Jonathan Martin
Marc Massot
Kartikeya Mayaram
Patrick C. McGeer
Rick McGeer
Ken McMillan
Amit Mehrotra
Robert G. Meyer
Robert S. Meyer
Trevor Meyerowitz
Giovanni De Micheli
Paolo Miliozzi
Sandra Moral
Rajeev Murgai
Amit Nandi
Amit Narayan
Arnit Narayan
A. Richard Newton
Yoshihito Nishizaki
Arlindo L. Oliveira
Ralf H. J. M. Otten
Davide Pandini
Claudio Passerone
Roberto Passerone
Claudio Passeronge
Yatish Patel
Claudio Pinello
Jan Rabaey
Anand Raghunathan
Rajeev K. Ranjan
Stephen Ricca
Howard S.

Richard L. Rudell
Jagesh V. Sanghavi
A. Sangiovanni-Vincentelli
Claudio Sansoè
Steve Sapiro
Larry Saunders
Hamid Savoj
Carl Sechen
Ellen Sentovich
Ellen M. Sentovich
Marco Sgroi
Henry Sheng
Narendra Shenoy
Thomas Shiple
Thomas Robert Shiple
Steven E. Shulz
Mário J. Silva
Kanwar Jit Singh
Vigyan Singhal
Kei Suzuki
Abdallah Tabbara
Bassam Tabbara
Johan Van Ginderdeuren
Iasson Vassiliou
Tiziano Villa
Yosinori Watanabe
Donald Webber
Ruey-Sing Wei
Jacob White
Jacob K. White
Wayne Wolf
Hormoz Yaghutiel
Alexandre Yakovlev
Guang Yang
Naeem Zafar
Stefano Zanella

Rifkin
Fablo Romeo
James A.
Rowson

↑ **Peer to Peer - Readers of this Article have also read:**

- Data structures for quadtree approximation and compression
  Communications of the ACM 28, 9
  Hanan Samet

- A hierarchical single-key-lock access control using the Chinese remainder theorem
  Proceedings of the 1992 ACM/SIGAPP Symposium on Applied computing
  Kim S. Lee , Huizhu Lu , D. D. Fisher

- 3D representations for software visualization
  Proceedings of the 2003 ACM symposium on Software visualization
  Andrian Marcus , Louis Feng , Jonathan I. Maletic

- Probabilistic surfaces: point based primitives to show surface uncertainty
  Proceedings of the conference on Visualization '02
  Gevorg Grigoryan , Penny Rheingans

- Efficient simplification of point-sampled surfaces
  Proceedings of the conference on Visualization '02
  Mark Pauly , Markus Gross , Leif P. Kobbelt

Software Timing Analysis Using HW/SW Cosimulation and Instruction Set Simulator

Jie Liu
Department of EECS
University of California
Berkeley, CA 94720
liuj@eecs.berkeley.edu

Marcello Lajolo
Dipartimento di Elettronica
Politecnico di Torino
Torino, ITALY 10129
lajolo@polito.it

Alberto Sangiovanni-Vincentelli
Department of EECS
University of California
Berkeley, CA 94720
alberto@eecs.berkeley.edu

Abstract

Timing analysis for checking satisfaction of constraints is a crucial problem in real-time system design. In some current approaches, the delay of software modules is precalculated by a software performance estimation method, which is not accurate enough for hard real-time systems and complicated designs. In this paper, we present an approach to integrate a clock-cycle-accurate instruction set simulator (ISS) with a fast event-based system simulator. By using the ISS, the delay of events can be measured instead of estimated. An interprocess communication architecture and a simple protocol are designed to meet the requirement of robustness and flexibility. A cached refinement scheme is presented to improve the performance at the expense of accuracy. The scheme is especially effective for applications in which the delay of basic blocks is approximately data-independent. We also discuss the implementation issues by using the Ptolemy simulation environment and the ST20 simulator as an example.

1. Introduction

Timing is one of the most important issues in real-time embedded system designs. The correctness of designs largely depends on the correctness of the interaction of hardware and software modules. Hardware/software cosimulation provides an integrated way to simulate this interaction, because it not only simulates the functionality but also simulates the delay of each module and the timing relations among the events. So it is essential for the simulator to use timing information for each module, especially software modules, which is as close to reality as possible. An incorrect delay may cause the simulation result to differ from the behavior of the implemented system.

Polis[1] is a hardware/software codesign environment for control dominated embedded systems. Polis is based on a formal model of computation called codesign finite state machine(CFSM). In Polis, systems are modeled as a group of communicating CFSM's, each of which is originally described in a formal language. e.g. Esterel[3]. In the simulations phase, both hardware and software modules are simulated in the Ptolemy environment.

Ptolemy[2] is a complete design environment for simulation and synthesis of mixed hardware-software embedded systems. In Ptolemy jargon, each functional block (a software or a hardware module) is called a star. Each star has one or more input and output ports. Stars talk to each other through links between ports that carry discrete events as FIFO queues.

A typical design flow in Polis is as follows (of course there will be feedback among different stages):

1. Create the system specification using synchronous-reactive system specification tools, such as Esterel.
2. Compile the source code and generate CFSM models.
3. Build simulation modules (stars) in the Discrete Event domain of Ptolemy.
4. Select target resources (microcontroller and real-time scheduler), assign each star as either hardware or software.
5. Run the simulation in Ptolemy, paying special attention to deadline violation and timing consistency.
6. If the simulation result is satisfying, synthesize the system into software and/or hardware.
7. Repeat more detailed simulation.

The software performance issue (clock cycles needed for a software module to execute) is involved in step 5 above.

Software performance has to be estimated in hardware/software codesign tools[4][5][10]. The advantages are small simulation overhead, ease of integration, and flexibility of porting to multiple microprocessors[8]. The disadvantage is the poor accuracy. For hard real-time embedded system, it is crucial to provide accurate timing information during the simulation phase.

Instruction Set Simulators (ISS) are software environments which can read microprocessor instructions and simulate their execution. Most of these tools can provide simulation results like values in memory and registers, as well as timing information (e.g. clock cycle statistics). Thus, ISS's provide a way to refine the timing calculation during the simulation phase.

We begin in section 2 with the analysis of existing soft-

ware timing estimation methods and their restrictions. In section 3, an interprocess communication architecture and a simple protocol are designed for integrating Ptolemy with ISS's. In order to improve performance, a cached timing refinement scheme is presented in section 4. An implementation example is given in section 5.

2. Related Approaches

Although the idea of performing a hardware/software cosimulation by combining RTL hardware simulation with cycle accurate instruction set simulators (ISS) is not new, a seamless integration of the two environments with high accuracy and good performance is still an unsolved problem.

In [9] a method which loosely links a hardware simulator with a software process is proposed. Synchronization is achieved by using the standard interprocess communication (IPC) mechanisms offered by the host operating system. One of the problems with this approach is that the relative clocks of software and hardware simulation are not synchronized. This requires the use of handshaking protocols, which may impose an undue burden on the implementation. This may happen, for example, because hardware and software would not need such handshaking since the hardware part runs in reality much faster than in simulation.

In [5] a method, which keeps track of time in software and hardware independently and synchronizes them periodically, is described. In [7] an optimistic and non-IPC approach for improving the performance of single-processor timed cosimulation are presented.

The biggest problem with all the current strategies is how to optimize the communication overhead required to synchronize the execution of the ISS and hardware simulator. Simply synchronizing the hardware and the software simulator in a lock-step fashion at every clock cycle would result in very limited performance due to the very low simulation speed obtainable. The other typical drawback of this approach is that the system has to be recompiled and resynthesized when the partition is changed.

The commercial tool Mentor-Seamless CVE [12] can offer a certain number of optimization techniques such as an instruction-fetch optimization that can directly fetch operations to the memory-image server, thus eliminating these cycles from the logic simulator workload. With a memory image server for memory read/write optimization, it can process operations 10,000 time faster than a logic simulator. This optimization is controlled by the user: early in the co-verification process, all memory operations should be directed to the logic simulator to debug all the operations of the memory sub-system. As the co-verification progresses, larger amounts of memory access can be optimized gradually, further speeding up the software execution.

The link with an ISS is not only applicable to hardware software cosimulation, but also useful for the general problem of embedded program timing analysis. For example, in [4] an approach which combines simulation with formal techniques is presented. They address the timing analysis problem by trying to approach each step in the analysis with the best methods currently known. They perform an architecture classification and a successive analysis and combine the classical approach of Instruction Timing Addition (ITA), in which the addition of the execution times in a basic block or in a path segment is computed, with a Path Segment Simulation (PSS), in which a cycle true processor model is used on a specific program path. Basically, they use the ITA approach for addressing the problem of data dependent instruction execution timing typical of some microcoded CISC architectures (e.g. multiplication) and PSS for all the other problems related to the impact of the architecture (pipelining, caching, superscalarity). The ISS is run only once on all basic blocks in the program and then the information collected is used with a formal analysis to determine the worst case execution time of the program.

Our approach is close to [4], but we determine which basic blocks are used at run-time by using a delay caching technique. Our approach is to leverage on an existing approach to time-approximate cosimulation[1], based on source-code estimation of execution time, and refine its precision by using an ISS. In particular, we do not require the designer to change the system specification. Hence we preserve the partitioning approach based on rapid interaction with the simulator, without the need to recompile or modify the specification (other than change implementation attributes for modules). The performance remains acceptable, with only a slight decrease in accuracy, thanks to a totally automated caching approach.

We also standardize the interface between the ISS and the system simulator, thus making the porting to a new ISS very easy.

3. Architecture and Protocol Design

The integration of Ptolemy and ISS should have the following properties:

- Supporting different ISS's
- Uniform interface
- Minimize code generation

To achieve these requirements, an interprocess communication architecture is provided. More precisely, a wrapper program is designed to glue Ptolemy and the ISS together.

The IPC architecture is shown in Figure 1. The wrapper

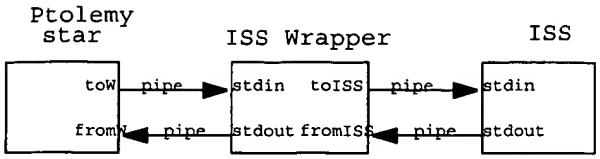


Figure 1. Interprocess Communication Architecture

program acts both as an interface unifier and a command translator. For each kind of ISS, a wrapper is built specifically so that it accepts the predefined ISS commands from Ptolemy, translates them into ISS syntax, and vice versa.

The syntax is defined to be concise enough so that the communication on each pipe is minimized. In other words, the commands adapt the CFSM simulation model to the ISS. The set of commands are listed in Table 1.

Table 1: Ptolemy-ISS Command Stack

Name	Syntax	Description
start	s <module> <?output?>	Initialize ISS, load binary code, set breakpoints at the beginning of the module and the corresponding output event emission point. If <output> is not set, the default end point is the end of the module.
write	w <variable> <value>	set a variable; variables could be CFSM states, event flags, star parameters, input values.
run	r	simulate up to a breakpoint
statistic	z	get the clock cycle statistic
quit	q	terminate the simulator.

The system works as follows:

❒In the setup phase of each star that uses an external ISS, it first checks if the wrapper has been started. If not, it forks a subprocess and executes the wrapper.

❒Each time the star gets fired in Ptolemy, it detects whether the delay needs to be refined (see section 5). If so, it sends the module information together with the variables to the wrapper.

❒After the wrapper has received all the information needed to run the simulator, it translates it into the ISS syntax, and sends it to the ISS.

❒The ISS loads the executable code of the corresponding star, sets the variable values and breakpoints, and executes the code up to the required breakpoint.

❒The wrapper waits until the ISS finishes the requested execution, and gets the clock cycle counting information.

❒The wrapper sends the clock cycle counting result to the Ptolemy star in a pre-defined format.

When this value is returned from the wrapper, the star adds it to the timestamp of the input event and gets the time-stamp of the output event. Then, one communication session is considered finished.

4. Cached Refinement Scheme

It is easy to imagine that, for a long run, if an ISS is used for every firing of a software star, the simulation could be quite slow. In order to speed up the timing refinement process and to keep the advantage of accurate clock cycle counting, we designed a cached refinement scheme, based on the properties of the s-graph and the discrete event semantics.

4.1. S-graph

An s-graph is a model of software execution that is used by the Polis cosimulation. It is a directed acyclic graph (DAG) with one source node called BEGIN and one sink node END. In s-graphs, there are two more types of nodes, ASSIGN and TEST. ASSIGN and BEGIN nodes have only one successor, and TEST nodes have two or more. An expression is associated with each TEST node, and according to the value of the expression (boolean or integer) the TEST node selects one of its children. An execution of an s-graph starts from the BEGIN node, traverses several nodes, until it reaches the END node. The sequence of nodes and edges during one execution forms a path of the execution. From the structure of the s-graph, it is easy to conclude:

Property 4.1: An s-graph can only have a finite number of paths, and for each execution the path is completely determined by the expression values at TEST nodes.

A node is called an EMIT node if its function is to emit an event to an output port of the CFSM. The time delay of emitting an output event is the time cost of the execution from the BEGIN node to the corresponding EMIT node. If we treat the END node as a special kind of event emission, the expressions at TEST nodes together with the final point of the path (EMIT or END node) uniquely specify a path to generate an event.

Generally speaking, the time delay of an event is not equal to the sum of the delays of each node along the path. For example, optimizations of the compiler, caching and pipelining can largely effect the delays. In particular, deep pipelining can overlap the execution of several basic blocks in the program. In addition, program and instruction fetches introduce dependences of the execution time of a path on the sequence of instructions/data fetches. We can thus have a great variance in the delay information for different executions of the path under different input stimuli and internal conditions. However, when we look at an entire path, the delay often varies in a relatively small range. In other words, when considering a medium-large level of granularity, most software modules have execution

delays that are approximately independent of past executions and depend mostly on input data. For this sort of applications, the delay of an execution path can be stored and be used the next time when the same path is traversed. Although this approximation reduces the accuracy, the simulation execution time can be dramatically improved.

4.2. The Ptolemy Discrete Event Domain

The semantics of Ptolemy DE domain[6] is that all signals (events) have two fields, a tag and a value. The tag is the timestamp, which is totally ordered among all the events. A star simply receives events from its the input ports and generates events to its output ports. The process of generating new output events can be divided into two parts: calculating the values and finding the timestamps. These two tasks are independent of each other, and can be done separately. So it is possible to use Ptolemy for behavior simulation and the ISS for timing. Furthermore, in the process of finding out the value of the event, the s-graph is executed from the BEGIN node to the proper EMIT node or END node. If all the predicates on this path are recorded, the internal path information is extracted. This unique internal path can then be used for caching timing information.

4.3. Cached Timing Refinement Scheme

A set of variables `iTrace`, one for each TEST node and termination point, are used to record the expression values on the TEST nodes. A table is created to store the timing information. The table has two fields; one of which, called the *key,* is encoded from `iTrace`; the other, called the *delay,* stores the delay value. During the firing of a star, the behavior model is first executed in Ptolemy. Then, the executed path of the s-graph is stored in `iTrace`. If this particular path has been traversed before, i.e. the same *key* is in the table, the corresponding *delay* is read and used as the delay for this execution. In this process, the ISS is not called. If there is no such *key* in the table, the ISS is called as described in section 3. The returned delay value is used for event delay calculation, and at the same time a new entry is added in the table.

The advantage of using this scheme is that for each internal path, the external ISS is called only once. So the number of Ptolemy-ISS communications is significantly reduced. It is obvious that this scheme is only appropriate for s-graphs with approximately constant delay for each path.

4.4. Stochastic Analysis

Due to the increasing complexity of the processors that are used for embedded applications, it is impossible to ignore the effects of caching and pipelining in the software part both in cosimulation and in static estimation. Moreover, a single execution of each basic block of the program is not sufficient to accurately characterize the block, because it neglects the interaction with previous, subsequent and preempting basic blocks.

Although stochastic measures are considered unacceptable for hard real-time embedded systems, we believe that applying a statistical analysis on the results obtained from linking Ptolemy with the ISS at an early design stage could improve the accuracy of our high-level trade-off evaluation method by automatically stopping the slow cosimulation when the measured variance of the delay in a path goes below a certain threshold. Alternatively, this provides the user with a measure of which module is particularly hard to characterize (due to a high variance of measured delays), in order to decide to continue simulations with the ISS, or use only cached delays.

Sometimes applications such as multi-media use processor architectures with a sophisticated non-standard instruction set. At present, compilers are not able to generate good code for that type of architectures and the user must manually optimize the code. In the Polis approach, this would result in an assembly code subroutine called inside Esterel modules. In this case, the modules cannot be directly simulated in the Ptolemy environment because the code cannot be compiled on a workstation. It is desirable to skip the execution of that function in Ptolemy and update, by using the ISS, not only the delay information but also the arithmetic/logic result of the computation.

Both activities outlined are still at a very preliminary stage, but the results seem quite promising.

5. Implementation

A prototype of this scheme has been implemented by using the Ptolemy and the ST20 Toolset[11]. Two parameters are added for each star to allow users to choose the delay calculation method--that of [10], cached refinement, and uncached refinement. A hash table is used to record and retrieve the path delay. The key of the hash table is encoded from the `iTrace` variables and the ID of an output port.The typical flow of this implementation is shown in Figure2.

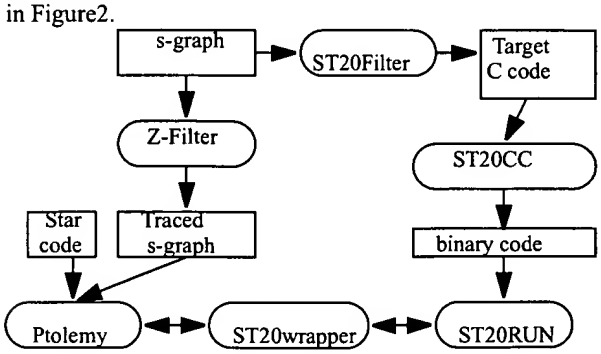


Figure 2. Implementation

Simple benchmarks are tested using these three approaches. COMPARE is a 2-input-2-output module which compares two integers A and B. If A>B, it emits output O1, otherwise it emits O2. ADDER is a floating point adder with one input A, one internal parameter B, and an output SUM. The results of using software performance estimation and the ISS are shown in Table 2 and Table 3 respectively, where `dctA` is the flag of detecting an input A; `A` stands for the value of input A; `output` is the termination point with 0 for END; `estimated` is the clock cycles obtained from the method of [10]; and `measured` is that from the ISS.

COMPARE is a module without function calls, where the compiler optimization make the delay of an entire path to be less than the delay of the sum of each node.

Table 2: COMPARE MODULE

state	dctA	dctB	output	estimated	measured
0	-	-	-	57	51
1	0	0	0	77	63
1	0	1	0	77	59
1	1	0	0	77	49
1	1	1	1	106	56
1	1	1	2	106	58
2	-	0	0	67	49
2	-	1	1	98	56
2	-	1	2	98	58
3	0	-	0	41	66
3	1	-	1	100	61
3	1	-	2	100	63

Table 3: ADDER MODULE

state	dctA	A	B	output	estimated	measured
0	-	-	0.1	0	120	114
1	0	-	-	1	45	50
1	1	0	0	1	887	392
1	1	0.1	0.1	1	887	755
1	1	2.3e12	5.3e8	1	887	790
1	1	0.123	0.987	1	887	855

Table 3 shows a module with two user defined functions; the delay of these functions are obtained from the statistic of sample runs with typical inputs. Also notice that the delay of a same path varies with the input data values.

The experiments are done on a SPARC20 workstation. For a simulation of 100 firings of COMPARE or ADDER, despite the time of starting the ISS (approximately 3 Sec.), the cached timing analysis is about 10 times slower than that of [10], but 1000 times faster than the non-cached exact timing analysis.

6. Conclusion

In this paper, an ISS-based timing refinement scheme is studied in the context of the Polis codesign approach. By using the ISS, some intrinsic problems of software performance estimation are solved. An open architecture is presented to adapt the differences among ISS's. Based on the properties of the s-graph and the DE semantics, a cached timing refinement scheme is studied. The result is that for s-graphs with approximately constant delay paths, the delay of one execution can be stored and reused the next time when the same path is traversed. A prototype has been implemented by using the Ptolemy and the ST20 simulation tools. The result shows that the timing analysis scheme can be seamlessly integrated into the Polis environment.

Acknowledgments

Thanks to Dr. Luciano Lavagno for his precious comments and suggestions during the working of this paper; also to Giovanni Bestente, Fabio Bellifemine, Antonio Bonomo and Giovanni Ghigo of CSELT, Torino, Italy for their help with defining the problem and outlining solutions

References

[1] F. Balarin, M. Chiodo, etc. "Hardware-Software Co-design of Embedded Systems", Kluwer Academic Publishers, 1997

[2] Ptolemy Home Page, "http://ptolemy.eecs.berkeley.edu"

[3] Esterel Home Page, "http://www.inria.tr/meije/esterel"

[4] R. Ernst, W. Ye, "Embedded Program Timing Analysis Based on Path Clustering and architecture classification", Proceedings of ICCAD, Nov. 97, PP 598-60

[5] K. ten Hagen, H. Meyr, "Timed and Untimed Hardware Software Cosimulation: Application and Efficient Implementation", Proceedings of Int. Workshop on Hardware-Software Codesign, Oct. 1993

[6] E. A. Lee and A. Sangiovanni-Vincentelli, "A Denotational Framework for Comparing Models of Computation", ERL Memorandum UCB/ERL M97/11, University of California, Berkeley, CA 94720, January 30, 1997.

[7] S. Leef, "Hardware and Software Co-Verification - Key to Co-Design", Electronic Design, September 15, 97, PP 67-71

[8] C. Passerone, L.Lavagno, etc. "Trade-off Evaluation in Embedded System Design via Co-simulation". Proceedings of ASP-DAC, PP 291-297, 1997.

[9] J. Rowson, "Hardware/Software Co-simulation", Proceedings of DAC, 1994, PP 439-440.

[10] K. Suzuki and A. Sangiovanni-Vincentelli, "Efficient Software Performance Estimation Methods for Hardware/Software Codesign", Proceedings of DAC, 1996, PP605-610.

[11] "ST20 'Osprey' Toolset: User Manual", SGS-THOMSON Electronics, March 1997

[12] S. Yoo, K. Choi, "Synchronization Overhead Reduction in Timed Cosimulation" Proceedings of Int. High Level Design Validation.

Uninterpreted Co-Simulation for Performance Evaluation of Hw/Sw Systems

Full text Publisher Site Pdf (1.03 MB)

Source **International Conference on Hardware Software Codesign** archive
Proceedings of the 4th International Workshop on Hardware/Software Co-Design table of contents
Page: 132
Year of Publication: 1996
ISBN:0-8186-7243-9

Authors Jean Paul Calvez
Dominique Heller
Olivier Pasquier

Publisher IEEE Computer Society Washington, DC, USA

↑ ABSTRACT

Performance modeling and evaluation of embedded hardware/software systems is important to help the CoDesign process. The hardware/software partitioning needs to be evaluated before synthesizing the solution. This paper presents a co-simulation technique based on the use of an uninterpreted model able to accurately represent the behavior of the whole system. The performance model includes two complementary viewpoints: the structural viewpoint which describes the functional structure, the hardware structure, the functional to hardware mapping, and the behavioral viewpoint which specifies the temporal evolution of each function or process. Attributes are added to the graphical model to specify the local properties of all components. The performance properties of the solution are obtained by simulation with VHDL. Software functions are executed according to the availability of an execution resource which simulates a microprocessor. This technique leads to rapidly obtain a lot of results by modifying appropriate parameters of the model, and so to easily scan the CoDesign space to decide on the best implementation. This modeling and estimation technique is fully integrated in a whole development process based on the MCSE methodology.

↑ INDEX TERMS

Keywords:
Hw/Sw systems, Performance evaluation, Co-Simulation, uninterpreted model

↑ Collaborative Colleagues:

Jean Paul Calvez: Charles Edmundson
Dominique Heller
Olivier Pasquier

Alan Wyche

Dominique Heller: Jean Paul Calvez
Olivier Pasquier

Olivier Pasquier: Jean Paul Calvez
Dominique Heller

Uninterpreted Co-Simulation for Performance Evaluation of Hw/Sw Systems

J.P. Calvez, D. Heller, O. Pasquier

IRESTE University of NANTES, La Chantrerie CP 3003 44087 NANTES Cedex 03
FRANCE, Email: jcalvez@ireste.fr

Abstract

Performance modeling and evaluation of embedded hardware/software systems is important to help the CoDesign process. The hardware/software partitioning needs to be evaluated before synthesizing the solution. This paper presents a co-simulation technique based on the use of an uninterpreted model able to accurately represent the behavior of the whole system. The performance model includes two complementary viewpoints: the structural viewpoint which describes the functional structure, the hardware structure, the functional to hardware mapping, and the behavioral viewpoint which specifies the temporal evolution of each function or process. Attributes are added to the graphical model to specify the local properties of all components.

The performance properties of the solution are obtained by simulation with VHDL. Software functions are executed according to the availability of an execution resource which simulates a microprocessor. This technique leads to rapidly obtain a lot of results by modifying appropriate parameters of the model, and so to easily scan the CoDesign space to decide on the best implementation. This modeling and estimation technique is fully integrated in a whole development process based on the MCSE methodology.

1: Introduction

In CoDesign, one major problem concerns the performance evaluation during the design step. Indeed, designers first have to define the appropriate functional architecture and then to find the partitioning and the allocation on the selected hardware. This means that the solution is deduced from the required performance constraints.

First of all, in order to answer correctly the design objective, one needs to consider the whole development life cycle and to base system developments on a complete design model and methodology. The work presented here is based on the use of the MCSE methodology [4] and specifically on the benefits of the functional model. Then, all along the design process, the selected solution has to be verified and evaluated in accordance to functional and non-functional requirements.

In order to avoid the late discovering of performances not met, the objective of the CoDesign method is to establish and maintain a strong link between the two concurrent developments: hardware and software. The two development branches result from the Hw/Sw partitioning. Deciding on an appropriate partition is therefore essential.

In this paper, we describe an efficient technique to evaluate performance properties of embedded Hw/Sw systems in order to correctly decide on partitioning and allocation according to performance constraints. Section 2 presents an important goal the designer is faced with. Section 3 describes the proposed CoDesign process. Section 4 briefly presents the uninterpreted performance model and the meaning of some attributes. Section 5 describes the co-simulation technique we are developing. Through an example, Section 6 explains the use of this model to extract performance properties and decide on the partitioning. Conclusions are drawn in the last section.

2: The partitioning goal in CoDesign

The final quality of systems that designers develop is mostly dependent on the development process. The first step is concerned with customer requirements which are then translated into functional and non-functional specifications. Performance constraints are one important category of non-functional specifications for Hw/Sw systems. From the specifications, designers have to decide on an architecture able to satisfy the application functionalities and performance constraints. The partitioning and the allocation of functionalities onto components are decided on during the CoDesign step [11],[16]. The last steps concern the implementation, the unit tests, the integration of all the parts, the tests and certification of the whole system with its environment.

Partitioning and allocation are strongly dependent on non-functional constraints: performances, timing constraints, cost, time-to-market, etc. One problem is to correctly elicit these requirements with the customer. Another problem we consider here is how to decide on the

Proceedings of the 4th International Workshop on Hardware/Software Co-Design (Codes/CASHE'96)
0-8186-7243-9/96 $10.00 © 1996 IEEE

partitioning. As a matter of fact, designers have to estimate and predict the performances of selected architecture(s) and compare them with the requirements. Later, during the synthesis of the solution which leads to the implementation step, more accurate information is available to refine the performance estimation and, if necessary, correct the design.

Performances qualify the behavior of the system relatively to observation criteria which may be external to the system (response time, throughput, etc.) or internal (utilization of a resource, bus throughput, etc.) [2],[7],[13]. Each kind of performance is called a performance index. Here we are concerned with the dynamic performances of real-time systems which are the most difficult to estimate and satisfy.

The estimation of system performances is usually done by analytical methods or simulation techniques [13]. In order to select the simulation technique for its capability to model transients, two types of models are possible: interpreted and uninterpreted. An *uninterpreted model* is a model for which the behavior is not dependent on the data values. It is the contrary for an interpreted model as it is the case for an algorithm or a state-based diagram. Therefore, an uninterpreted model is a more abstract model or is an abstraction of an interpreted model obtained by removing the data or information values. The effect of these data values are abstracted and replaced by attributes. For example, the attribute *Execution Time* replaces the execution duration of a sequence of statements on a processor, the attributes *Size* and *Id* replace the content of a message. Few performance models and tools exist to evaluate the dynamic performances of any kind of systems [1],[14].

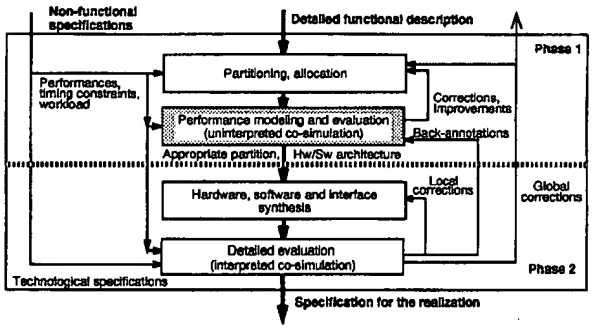
In CoDesign, an accurate estimation of the temporal properties of a solution needs to simulate the hardware part and the software part together. Since a microprocessor is used to run several processes or tasks, its properties and the task-scheduling policy mainly define the global system behavior. The time scale is not the same either: > 1 µs for the software, < 100 ns for the hardware. Therefore a *co-simulation* is necessary.

To help designers during the partitioning phase, we propose to use a performance model which is an uninterpreted model to represent the hardware and software organization and behavior of the solution. Properties are extracted by co-simulation of this model, which means the simulation of the software and the hardware together. This technique is much faster than using an interpreted model and easily allows to study the influence of some specific parameters. Generic architectures may also be studied.

3: Presentation of the method

So as to correctly master the partitioning and allocation in order to find the most appropriate mapping of the functional description onto a hardware architecture, the CoDesign process we propose is depicted in Figure 1. For more details on the global design process, the reader can refer to [4], [9]. In our approach, before partitioning, the designer needs to correctly delimitate the critical parts of the project, to design a functional solution, to specify the performance requirements and the system workload conditions, to define a detailed functional solution including the geographic partitioning constraints and the physical interfaces.

The CoDesign stage is decomposed into two phases, in each one a verification by co-simulation enables to decide on corrections or to continue.



-Figure 1 - The CoDesign process with performance estimation.

The partitioning and allocation can be based on various methods: automatic, semi-automatic, interactive [16]. Since the input functional description is conform to the MCSE methodology, in [5],[9] we suggested to follow an interactive coarse-gain partitioning procedure driven by the designer who can easily decide on an appropriate choice for each function.

The uninterpreted performance model presented in the next section is then easy to obtain. The structural model results from the composition of the functional structure and the hardware architecture according to the mapping. The behavioral model of each function is an abstraction of the algorithmic behavior. Attributes and parameters specify the properties of all components. The workload of the system is used to define the context of the simulation. The performance indexes are used for selecting the results to observe.

In the second phase, when an appropriate partition is reached, the functional description, the hardware architecture and the mapping are used to obtain the hardware and software descriptions by synthesis [12]. Both descriptions are used for a final verification by a detailed interpreted co-simulation. A back-annotation of execution times is also possible to enhance the performance model.

This process allows to follow a smooth incremental design path with a better integration of performance mastering. In this way the correction or improvement feedback loop is shorter. As a matter of fact, without the

Proceedings of the 4th International Workshop on Hardware/Software Co-Design (Codes/CASHE'96)
0-8186-7243-9/96 $10.00 © 1996 IEEE

uninterpreted performance modeling and co-simulation phase, the verification of performance satisfaction is possible only after the complete synthesis of the solution and a detailed co-simulation which needs more time.

4: Presentation of the model

The conceptual model of MCSE [4] includes two views, each corresponding to a specific aspect of the solution:

- the *functional model* (hierarchical and graphical model) describes a system by a set of interacting functional elements (organizational dimension or functional structure) and the behavior of each of them.
- the *executive model* describes the architectural structure based on active components (microprocessors, specific processors, analog and digital components) and interconnections between them.

These two views, when separately considered, are not sufficient to completely describe the solution of Hw/Sw systems. It is necessary to add the *mapping* between the functional and the executive viewpoints, defining an integration or allocation also called configuration.

The functional model, located between models appropriate to express specifications and models to describe the architecture, is suitable to represent the internal organization of a system by explaining all necessary functions and couplings between them according to the problem viewpoint. Designing with this method leads to an internal technology-independent solution. All or part of the description may be implemented either in software or in hardware. Therefore, this model is interesting as a specification input for a Hw/Sw CoDesign method based on a coarse-grain partitioning

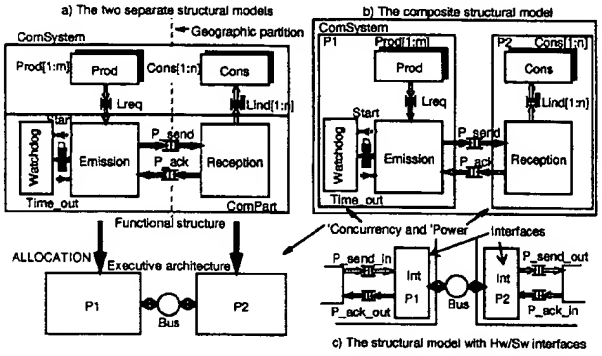
We enhanced this model according to two complementary and orthogonal viewpoints to extend its usefulness to performance modeling:

- the *organizational viewpoint* (structural model) which describes the system by a hierarchical structure including the above functional and executive structures.
- the *behavioral viewpoint* for each function or component, which specifies the set of operations and their total or partial time ordering. This is an uninterpreted model of the function.

In the next sections we briefly introduce the two viewpoints. More information on the performance model and its use can be found in [6], [10]. In [6] this model was used to analyze the performance properties of a real-time video server.

4.1: The structural model

The meaning of the structural model is extended by considering both the functional meaning (function, event, shared variable, port) and the executive meaning (processor, signal, shared memory, communication node). Thus, it is possible to represent both structures - functional and executive - with the same graphical model, and so to describe the complete architectural solution with the partitioning and allocation (functional to executive binding). Figure 2 illustrates the concept. On the left hand side, two structures and the partitioning and allocation are depicted. On the right hand side, only one structure represents the same solution.



-Figure 2 - Structural model for performance modeling.

The example considered here is a simplified communication system *ComSystem* for message transfer between producers *Prod[1:m]* and Consumers *Cons[1:n]*. The function *Emission* has to send each message of *LReq* to its corresponding function *Reception* through the port *P_Send*. To guarantee a correct transfer, each message has to be acknowledged via the port *P_Ack*. *Emission* uses a watchdog function to limit the waiting duration of the acknowledgement.

The executive structure is composed of two processors linked by a node representing a bus. Each processor can be characterized by two attributes: 'Concurrency (the number of functions it can execute simultaneously) and 'Power (relative CPU speed value). The bus can be specified by its concurrency (number of simultaneous accesses), its send and receive times for each message. Here the chosen allocation is simple to understand since it is based on the geographic partition constraint.

The objective of the performance model (structural viewpoint) on the right hand side is to represent the two structures and the allocation with only one model. Figure 2-b depicts such a composite or combined structural model. Starting from the functional structure, each processor *P1* and *P2* is added as an encapsulation of the set of functions that each processor has to execute. This operation corresponds to a graph restructuration which is called folding: a group of nodes in a graph is selected to form a new node composed of the subnodes previously selected. In this way, *P1* and *P2* have in fact the meaning of a function with nevertheless the two specific properties of a processor:

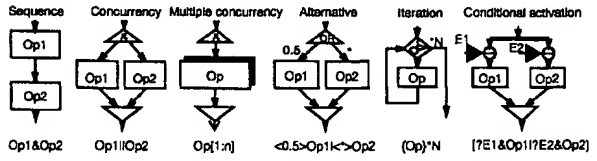
Proceedings of the 4th International Workshop on Hardware/Software Co-Design (Codes/CASHE'96)
0-8186-7243-9/96 $10.00 © 1996 IEEE

'Concurrency and 'Power. In this structure the inter-processor link through the node *Bus* is not considered for simplification. The temporal properties of the link are abstracted and integrated into the 2 relations by *P_Send* and *P_Ack*. If the above structural model is considered too abstract, it is possible to keep the *Bus* link. In that case, interfaces (which are functions) between functions and processors must necessarily be added (Figure 2-c).

4.2: The behavioral model

The behavioral viewpoint of an active component is orthogonal to the structural one. A behavioral model, which is hierarchical, graphical and uninterpreted, is described according to the vertical axis representing the temporal evolution and the horizontal axis describing the data or information flow (transactions).

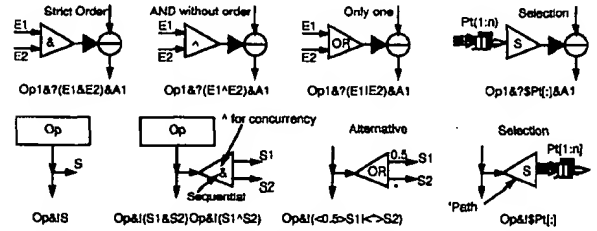
The temporal description is based on five kinds of composition: sequence (&), alternative (|), concurrency (||), iteration ({-}), conditional (guarded) activation ([?-|?-]). Figure 3 gives the graphical notation for each of them (vertical axis). Exclusive or concurrent evolutions are drawn as parallel branches. Complex internal behaviors result from dynamic instanciations of activities and activity refinements. An activity considered elementary is called an operation.



-Figure 3 - Representation of composition rules.

The modeling of performance implies the notion of execution time for operations and exchanges in order to extract the global temporal properties of a system from its local temporal properties. Therefore, an execution time (attribute *'Time*) is added to each operation.

To express interaction rules between vertical branches Temporal dependencies, (i.e. synchronizations, communications) and inputs and outputs are represented horizontally. An activation condition is elaborated from available inputs. The generation of outputs or internal data are actions executed after operations. Composition operators for interactions are represented in Figure 4.



-Figure 4 - Notation for function or activity interdependencies.

The actions concern the generation of an information item or of an event through the outputs of the component including the activity or towards other internal activities. For a shared variable, the action concerns a reading or a writing operation. The Alternative symbol leads to produce only one output. In this case, a rule (attribute) must be specified to define the output concerned: determinism, random. The Sequence symbol defines the order of reception or generation. The Selection symbol allows to specify which input or which output in a set or in a vector is concerned. The attribute *'Path* is used for that purpose. The behavioral model is illustrated by the example given further in Section 6.

An information item or a transaction is defined with a set of predefined and user carried attributes. All attributes are useful to control the temporal evolution of functions, activities and operations of the whole structure.

4.3: Attributes and parameters

To extract results from this uninterpreted performance model, attributes must be added to the above graphical notation.

The predefined attributes of the structural model concern each active component and each relation component. For a function, a processor and even a system, (i.e. an active structural component), we have selected the following attributes:

- *'Power*: floating-point value, (1 by default)
- *'Concurrency*: a positive integer, (1)
- *'Policy*: (PSP, PSD, NPS, TS), (PSP means Preemptive Scheduling based on Priority)
- *'Overhead*: a time, (0)
- *'Priority*: an integer, (1 for the lowest priority)
- *'Deadline*: a time, (0).

These attributes are useful to evaluate the properties of an architecture. The attribute 'Power simulates the power of a computer unit; execution times of all included active elements are scaled by this coefficient. It simulates the clock speed of the computer. The attribute 'Concurrency is useful to simulate a component having a limitation of running resources. With it, it is easy to simulate a monoprocessor or multiprocessor type of component. By changing the attribute 'Policy, it is also easy to experiment with the influence of different policies and compare them. The attribute 'Overhead is useful to simulate the time needed for task context switching. The last two attributes 'Priority and 'Deadline are used to select the most urgent task to run (only one of the two values is used according to 'Policy).

Each kind of relation components is also specified by a set of attributes. Here, due to lack of space, we only give the selected attributes of a port:

- *'Policy*: (Fifo, Priority), (Fifo)
- *'Concurrency*: a positive integer, (1)

Proceedings of the 4th International Workshop on Hardware/Software Co-Design (Codes/CASHE'96)
0-8186-7243-9/96 $10.00 © 1996 IEEE

- *'Capacity*: max number of messages, ≥0, (1)
- *'Write*: a time, (0)
- *'Read*: a time, (0).

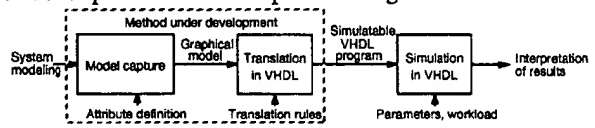
The predefined attributes of the behavioral model are: *'Time* for operation durations, *'Size* for the size of data or information items, *'Path* to specify a path through a selection operator, *'Cond* for a conditional loop, *'Id* for the identification of a function or an activity.

In general, the value of an attribute is dynamic and is defined by any mathematical expression including constant values, parameters, other attributes, the current time, mathematical and probabilistic functions.

The resulting model is an uninterpreted one. Notice also that several models may specify an active component at the same time. This means that during the top-down design process the behavioral model is a specification from which it is easy to deduce an equivalent structural solution.

5: Co-simulation and result extraction

Our performance modeling technique and the corresponding simulation method are an integral part of a set of tools we have been developing as a help to the MCSE methodology. The performance model has to be simulated to be usable for CoDesign as a help to decide on partitioning and allocation. Two techniques are possible: use of a specific simulator developed for the proposed model, translation of the model into a language for which a simulator already exists. In this latter case, the model is translated into an executable description. We are currently considering two techniques: translation into VHDL and then simulation to extract appropriate characteristics, translation into C++ and execution. The process under development to evaluate performance is depicted in Figure 4.



-Figure 5 - Process for performance evaluation.

Designers first have to define the appropriate performance model according to what they want to evaluate. The model is captured with graphical tools and the attributes of all the elements are added. The graphical model is then automatically translated into a simulatable VHDL program according to translation rules. The simulation VHDL model with defined parameters and an appropriate simulation of the workload of the system generates events and data which are interpreted to obtain the results.

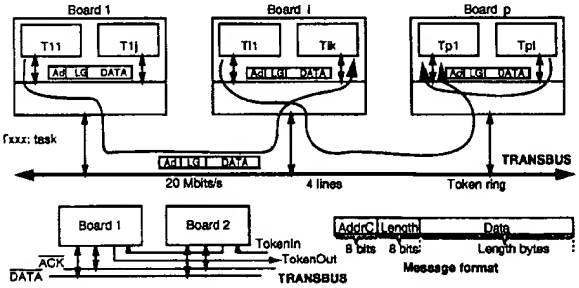
The performance analysis of the event trace leads to estimate the properties of the solution during the design step and to select the best solution and parameters [7].

VHDL is very efficient to describe and simulate concurrent functions and multiple instanciations with generic parameters. The simulation allows to extract various characteristics of architectures to evaluate their costs and performances. High-level descriptions are also very easy to describe and test in the form of uninterpreted models. Generic parameters are an efficient way to specify the behavior of all types of components of the model.

But for hardware/software co-simulation, we have observed some limitations due to the fact that VHDL was conceived to describe circuits rather than systems. Probably the main limitation of VHDL for our goal is the lack of an external process suspension including freezing the process time to simulate a multiple function processor sharing. Further details on the translation rules into VHDL are described in [6], more specifically the execution of several functions onto a limited processing resource.

6: An illustrative example

The case study we have chosen to illustrate our approach is described in [5],[9]. The required goal is to design and prototype a distributed communication system obtained by assembling many similar boards. On each board, producers have to send short messages or packets (256 bytes max.) to consumers located on the same board or on other boards. Producers and consumers are software tasks. A 20 Mbits/s serial bus called TransBus [3] is used to interconnect the boards. The system requirements and the bus specification are shown on Figure 7. Each message includes: the address of the consumer, the length of the data part and then the data.



-Figure 6 - Requirements of the communication system.

The bus access management is based on a hardware token ring. At any moment, only one board must own the token. The token is implemented as a boolean signal and all the boards are wired as a circular shift register. Only the token owner can send a message if needed and then pass on the token to its neighbor.

6.1: The problem to be solved

The designer's objective is to correctly define and implement a board according to performance requirements. A generic architecture is quite simple to imagine. In [9], we described an architecture based on a microprocessor, an FPGA and a shared memory. It is easy here to find that a

Proceedings of the 4th International Workshop on Hardware/Software Co-Design (Codes/CASHE'96)
0-8186-7243-9/96 $10.00 © 1996 IEEE

strict minimum of hardware is necessary to satisfy the 20 Mbits/s transmission rate and the bit protocol imposed by the Transbus. In the rest of the paper, we suppose the existence of this hardware part to implement the bus interface. It includes a parallel to serial convertor to transmit each byte and the reverse for the reception of each byte.

The problem here is to determine the remainder of the solution. The first step consists in defining the functional solution. This means identifying all processes and relations between them. The next step consists in defining the partitioning and the allocation. But to do so, it is necessary to have quantitative information on the required performances and on the performances estimated according to the selected functional design and generic hardware architecture. To obtain this information, a model of the solution is needed.

Rather than developing a complete interpreted model for both the software part and the hardware part and co-simulate it, we show in this example that it is relatively easy to estimate various performance indexes on different implementations with a generic uninterpreted model and a co-simulation of it. In the next sections, we describe the functional model, the behavioral model, the various results obtained by co-simulating the hardware and the software at a macroscopic level but sufficiently detailed to rapidly observe interesting results.

6.2: The functional model

To design this communicating system, it is necessary to take into account the decomposition of the system into a set of boards and the interconnection bus (the geographical distribution of the application). This task is well done by applying the specification and functional design steps of the MCSE methodology. The result of geographic partitioning and introducing the physical interface is described in Figure 7-a which presents the complete detailed functional solution of each board which satisfies these technological constraints. The transbus is here modelled (abstracted) in Figure 7-b by a vector of events *Token[1:k]* to represent the token ring and a vector of ports *TB[1:k]* to describe the behavior of the message transfer between each pair of boards.

Each message produced is sent by a producer *Prod[i]* to the function *Routing* through its port *Treq[i]*. The address field is used by *Routing* to determine if the designate consumer is local (same board) or distant. For each distant communication, the function *EmissionMess* sends each message from *Lreq* to the addressee board through the port *TB[addressee]*. Since only one board at one and the same time must access the TransBus, *EmissionMess* first has to request the token (event *EmisReq*) and wait for it (event *TokenOk*). When the message sending is finished, the event *EmisEnd* releases the token which is then sent to its neighbor (*Token[i+1 mod K]*). The function *ReceptionMess* receives each message which concerns the board and sends it to the function *Dmux* through the port *Lind*. *Dmux* sends each message to the addressee consumer.

6.3: The behavioral model

The behavior of each function in an uninterpreted form is given in Figure 7-c according to the notation described in Section 4. Attributes are added to the graphical model to specify the complete behavior. To understand the notations,

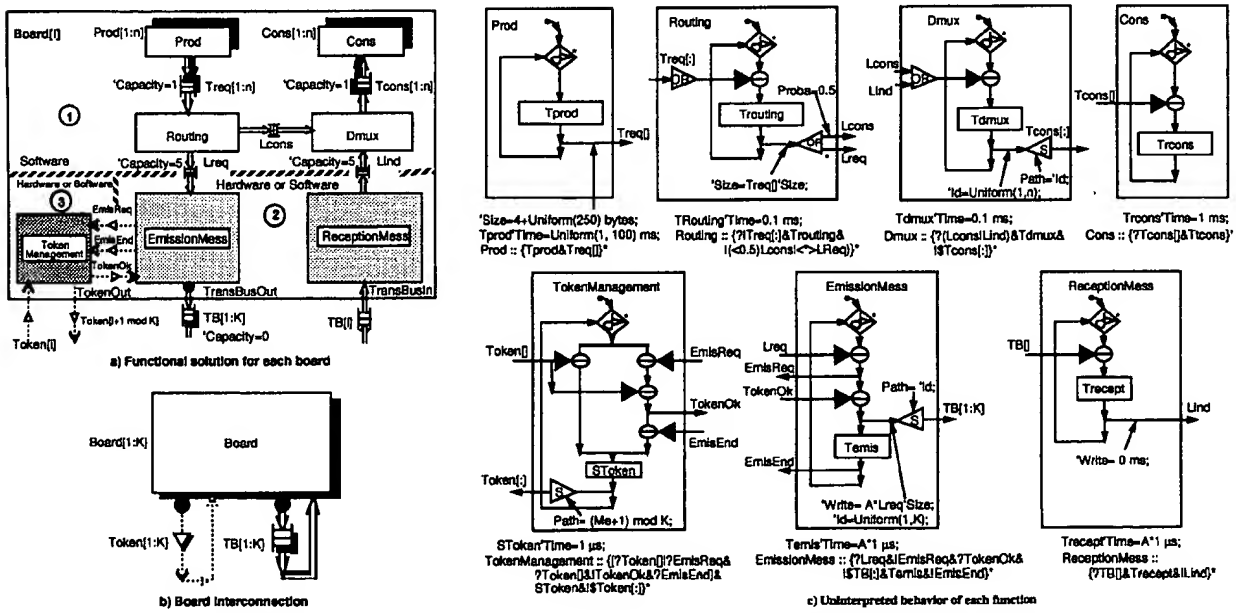


Figure 7- Functional structure for one board and the uninterpreted behavior of each function.

Proceedings of the 4th International Workshop on Hardware/Software Co-Design (Codes/CASHE'96)
0-8186-7243-9/96 $10.00 © 1996 IEEE

Treq[] means the port of the same index as the producer in the vector *Treq[1:n]*. *Treq[:]* designates the complete vector. This notation is interesting for generic components.

Producers and Consumers are very simple cyclic processes. The size of each produced message is a random value (function *Uniform*). The time interval between two successive messages is defined by the execution time of the operation *Tprod* (attribute 'Time). This is an easy way to specify the system workload for this example.

The function *Routing* receives messages from all ports in *Treq[1:n]* and simulates the routage of a part of them to the port *Lcons* (local messages). The attributes *Proba* and * (which means Else) are used to specify the selected output. Each generated message includes the attribute 'Size of the input message. *Dmux* receives messages from the ports *Lcons* and *Lind* and routes them to the consumer whose identifier is defined by the included attribute 'Id.

The function *TokenManagement* is in charge of the bus allocation toward only one board. *EmissionMess* receives a message from *Lreq* and then asks for the token. The transmission of each message on Transbus is modelled by sending it in the port of *TB[:]* whose index is equal to the attribute 'Id defined as a random index of a board. The transmission time is specified by the attribute 'Write which uses the size coming from the received message and the parameter *A* defining the time to transmit each byte. The function *ReceptionMess* is a cyclic process waiting for each message in its port *TB[]* and then sending it to the port *Lind*.

6.4: Co-simulation and results

The objective of this example is to show that the model described above leads to evaluate the main performances of different implementations of the functional solution. To enhance the CoDesign approach described in [5], for our example, we have experimented 3 different implementations. In Figure 8, the functional description is decomposed in 3 areas: area (1) includes functions compulsorily implemented in software, area (2) includes the two transmission and reception functions on the TransBus, and area (3) includes only the management of the token.

The question is: how are the performances modified when the functions in areas (2) and (3) are mapped onto hardware or software? To answer this question, we have simulated the uninterpreted model described above. Beforehand, it is important to correctly identify the system workload and the appropriate results expected in order to decide for the best implementation.

Concerning the system workload, we have stimulated the communication part of the system (emission and reception on the TransBus) with producers permanently sending messages of random size to distant random boards. In this case, *Tprod'Time* = 0 ms and in the function *Routing*, *Proba* = 0 (no local transmission). A consumer is also supposed to spend at least 1 ms to exploit each input message. The servo-control of producers to consumers is obtained by the capacity of each port (attribute 'Capacity). We have chosen: capacity of *Treq[i]* and of *Tcons[i]* = 1, capacity of *Lreq* and of *Lind* = 5. The correct simulation of TransBus is obtained with *TB[:]'Capacity* = 0, which means a rendez-vous between the sender *EmissionMess* and the receiver *ReceptionMess* connected to the port used.

Concerning the results to evaluate, we consider the following as representative of the efficiency of the communication system:

- the latency of a message from the producer to the consumer,
- the throughput on the TransBus,
- the utilization ratio of the processor running the software on each board.

Because of the random character of behavior of the model, the 3 results are evaluated as the average of all boards and all producers and when the steady state is reached (# 0.1 s observed by simulation).

The interest of the co-simulation is to study the influence of different generic parameters of the system. Therefore, we have varied the number of boards (3, 6, 9) and the number of producers and consumers (generic parameter n).

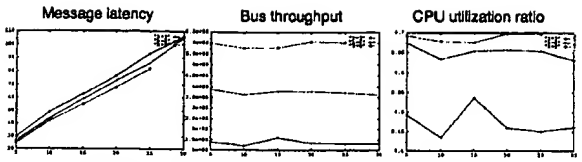
-A- Maximum hardware (areas (2) and (3) in hardware)

To obtain appropriate results, it is necessary to know the time needed to send each byte on TransBus when *EmissionMess* and *ReceptionMess* are implemented in hardware. The value is taken from [5] where we described the solution. Another direct means is to consider the TransBus protocol at the bit level: 11 bits x 50 ns # 0.7 µs. Therefore we have chosen A = 0.7 µs to represent the speed of the hardware. The time for *TokenManagement* is selected to 1 µs.

All software functions of a board are implemented on the same processor. To do that, a function named *Processor* is added which includes all the software functions. This function simulates a resource with a concurrency degree of 1, which means that only one included function can be active at one and the same time. Two interesting attributes define such a function: its concurrency, and its power (equal to 1 here). 'Power is interesting to modify the execution speed of all software functions and study its influence. The scheduling policy is also to be defined. The attribute 'Priority of each software function is used for that purpose. Here the priority is the highest for *Dmux*, then *Routing*, then *Cons[1:n]*, and the lowest for *Prod[1:n]*.

The results are given in Figure 8. K identifies the number of boards.

Proceedings of the 4th International Workshop on Hardware/Software Co-Design (Codes/CASHE'96)
0-8186-7243-9/96 $10.00 © 1996 IEEE

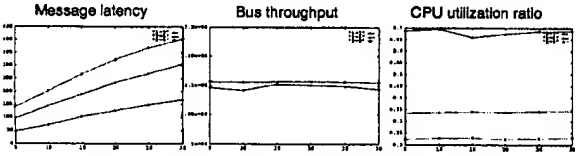


-Figure 8 - Results for the maximum in hardware.

The message throughput on each board is constant and equal to the size average of all the produced messages (129), each being consumed every 1 ms by a consumer. The bus throughput is not dependent on n but on K. The message latency increases with the number of producers and consumers because each CPU is shared by all of them. The CPU utilization rate is relatively low for K=3, because the bus is not properly used.

-B- All functions in software

All the functions are added inside the function *Processor*. The time needed to send each byte on TransBus is now A=7µs. The execution time chosen for the operation *SToken* is 20 µs which is the time needed for a CPU on receiving an interrupt. The results are given in Figure 9.



-Figure 9 - Results for all functions in software.

The bus throughput is now constant and a little lower than the previous case because 20 µs are added between two successive messages. The latency and the CPU utilization rate are similar.

7: Conclusion

In this paper, we have described a performance model and a co-simulation technique to help designers for system partitioning and allocation while developing embedded hardware/software systems. The model is of uninterpreted type, this means that it represents the whole solution at an abstract level but accurate enough to evaluate the system properties. Because of this type, the simulation is faster than a complete hardware/software interpreted model. The performance evaluation is based on a VHDL simulation; the VHDL program is obtained by a systematic translation of the graphical performance model and attributes according to specific translation rules.

A graphical tool and the automatic VHDL program generator is under development. We are also experimenting with a translation into C++ to obtain the performance results. The resulting method and the tool we are currently developing are fully integrated in the complete MCSE system-level methodology.

References

[1] J. Aylor, R. Waxman, B.W. Johnson, R.D. Williams, The integration of performance and functional modeling in VHDL, in "Performance and Fault Modeling with VHDL", J.M. Schoen, Editor, Prentice-Hall, New Jersey, 1992, pp 22-145

[2] R. Bordewisch, W. Föckeler, B. Schwärmer, F-J. Stewing, Non-Fonctional Aspects: System Performance Evaluation, In "Systems Engineering. Principles and Practice of Computer-Based Systems Engineering", Editor B. Thome, John Wiley, 1993, pp 223-271

[3] J.P. Calvez, O. Pasquier, A TRANSputer interconnection BUS for hard real-time systems, Transputer'92 Besançon France IOS Press, May 20-23, 1990, pp 273-283

[4] J.P. Calvez, Embedded Real-time Systems. A specification and Design Methodology, John Wiley, 1993

[5] J.P. Calvez, D. Isidoro, A CoDesign experience with the MCSE methodology, Proceedings of the Third International Workshop on Hardware/Software CoDesign, Grenoble, France, Sept 22-24, 1994, pp 140-147

[6] J.P. Calvez, D. Heller, O. Pasquier, System performance modeling and analysis with VHDL: Benefits and limitations, VHDL-FORUM EUROPE Conference, IRESTE, Nantes, France, April 24-27, 1995

[7] J.P. Calvez, O. Pasquier, Performance Assessment of Embedded Hw/Sw Systems, ICCD'95, International Conference on Computer Design, Austin, Texas, October 2-4 1995

[8] J.P. Calvez, A System Specification Model and Method, Current Issues In Electronic Modeling, Issue #4: Modeling of System Abstraction, Kluwer Academic Publishers, 1996

[9] J.P. Calvez, A CoDesign Case Study with the MCSE Methodology, To appear in the journal "Design Automation of Embedded Systems", Special issue on "Embedded Systems Case Studies", Kluwer Publisher, 1996

[10] J.P. Calvez, A System-level performance model and method, Submitted to "Current Issues In Electronic Modeling", Issue #6: Meta-modeling: Performance, Software and Information Modeling, Kluwer Academic Publishers, 1996

[11] D.D. Gajski, F. Vahid, S. Narayan, J. Gong, Specification and Design of Embedded Systems, Prentice Hall, Englewood Cliffs, New Jersey, 1994

[12] R.K. Gupta, G. De Micheli, Hardware-Software cosynthesis for digital systems, IEEE Design & test of computers, Sept. 1993, pp 29-41

[13] R. Jain, The Art of Computer Systems Performance Analysis, John Wiley, 1991

[14] SES/workbench: a multilevel design environment for modeling and evaluation of complex systems, Scientific and Engineering Software, Inc., August 1989

[15] D.E. Thomas, J.K. Adams, H. Schmit, A model and methodology for Hardware-Software Codesign, IEEE Design & test of computers, Sept. 1993, pp 6-15

[16] W.H. Wolf, Hardware-software Co-Design of embedded systems, Proceedings of the IEEE, Vol 82, No 7, July 1994, pp 967-989

Proceedings of the 4th International Workshop on Hardware/Software Co-Design (Codes/CASHE'96)
0-8186-7243-9/96 $10.00 © 1996 IEEE

How standards will enable hardware/software co-design

Full text Pdf (159 KB)

Source **International Conference on Hardware Software Codesign** archive
Proceedings of the seventh international workshop on Hardware/software codesign table of contents
Rome, Italy
Pages: 211 - 212
Year of Publication: 1999
ISBN:1-58113-132-1

Authors Mark Genoe
Chris Lennard
Joachim Kunkel
Brian Bailey
Gjalt de Jong
Grant Martin
Kamal Hashmi
Shay Ben-Chorin
Anssi Haverinnen

Sponsors IEEE-CS : Computer Society
IFIP : International Federation for Information Processing
SIGSOFT: ACM Special Interest Group on Software Engineering
SIGDA: ACM Special Interest Group on Design Automation

Publisher ACM Press New York, NY, USA

↑ INDEX TERMS

Primary Classification:
C. Computer Systems Organization
↳ **C.1** PROCESSOR ARCHITECTURES

General Terms:
Documentation, Standardization

↑ Collaborative Colleagues:

Brian Bailey:	Dan Beece Shay Ben-Chorin Geoff Bunza Rick Chapman Robert Cooley Michel Courtoy Moses Dejong John Fogelin Masahiro Fujita Dan Gajski	Mark Genoe Joost Geurts Rajesh Gupta Lynda Hardman Kamal Hashmi Anssi Haverinnen Willis Hendley Marion Kenefick Kurt Keutzer Joseph A. Konstan	Joachim Kunkel Chris Lennard Sharad Malik Grant Martin Amr Mohsen Richard Moseley Daya Nadamuni John O'Leary Carl Pixley Shishpal Rawat	Jim Rowson Lloyd Rutledge Sandeep Shukla Gary Smith Fabio Somenzi Gjalt de Jong Jacco van Ossenbruggen
Shay Ben-Chorin:	Brian Bailey Mark Genoe Kamal Hashmi Anssi Haverinnen Joachim Kunkel Chris Lennard Grant Martin Gjalt de Jong			
Mark Genoe:	Brian Bailey Shay Ben-Chorin Luc J. M. Claesen Kamal Hashmi Anssi Haverinnen Joachim Kunkel Chris Lennard Hugo De Man Grant Martin Frank Proesmans	Paul Vanoostende Eric Verlind Gjalt de Jong Geert van Wauwe		
Kamal Hashmi:	Brian Bailey Shay Ben-Chorin Mark Genoe Anssi Haverinnen Joachim Kunkel Chris Lennard Grant Martin Gjalt de Jong			
Anssi Haverinnen:	Brian Bailey Shay Ben-Chorin Mark Genoe Kamal Hashmi Joachim Kunkel Chris Lennard Grant Martin Gjalt de Jong			
Joachim Kunkel:	Brian Bailey Shay Ben-Chorin	Anssi Haverinnen Rick Hetherington Chris Lennard	Frank Schirrmeister Steven E. Schulz Diederik Verkest	

	Dennis Brophy Raul Camposano Clifford E. Cummings Simon Davidman Mark Genoe Abhijit Ghosh Richard Goering Kamal Hashmi	Larry Lerner Oz Levia Stan Liao Grant Martin Jan Rabaey Davoud Samani John Sanguinetti	Gjalt de Jong	
Chris Lennard:	Brian Bailey Shay Ben-Chorin Mark Genoe Kamal Hashmi Anssi Haverinnen Joachim Kunkel Grant Martin Gjalt de Jong			
Grant Martin:	Brian Bailey Shay Ben-Chorin Henry Chang Rick Chapman Rong Chen Larry Cooke John Fogelin Mark Genoe Thorsten Grötker Kamal Hashmi	Anssi Haverinnen Merrill Hunt R. Peter Hypher Kurt Keutzer Joachim Kunkel Luciano Lavagno Chris Lennard Jean Louis-Guerin Sharad Malik Radu Marculescu	Andrew J. McNelly Daya Nadamuni Jan Rabaey Alberto Sangiovanni-Vincentelli Frank Schirrmeister Bran Selic Marco Sgroi Sandeep Shukla Gary Smith Cadence Design Systems	Lee Todd Gjalt de Jong
Gjalt de Jong:	Brian Bailey Shay Ben-Chorin Francky Catthoor Kris Croes Hugo De Man Hugo J. De Man Mark Genoe Pete Hardee Kamal Hashmi Anssi Haverinen	Anssi Haverinnen Tilman Kolks Joachim Kunkel Chris Lennard Christopher K. Lennard Bill Lin Grant Martin Miguel Miranda Alex Niemegeers Patrick Schaumont	Paul Six Peter Slock Steven Vercauteren Bill Lin Carl Verdonck Diederik Verkest Eric Verlind Sven Wuytack Chantal Ykman-Couvreur Julio L. da Silva Julio Leao da Silva	Hugo de Man

↑ **Peer to Peer - Readers of this Article have also read:**

- MBONE: the multicast backbone
  Communications of the ACM 37, 8
  Hans Eriksson

HOW STANDARDS WILL ENABLE HARDWARE/SOFTWARE CO-DESIGN

Mark Genoe (Chair – Alcatel); Chris Lennard (Cadence) ; Joachim Kunkel (Synopsys) ;
Brian Bailey (Mentor Graphics); Gjalt de Jong (Alcatel) ; Grant Martin (Cadence);
Kamal Hashmi (Fujitsu – ICL), Shay Ben-Chorin (National); Anssi Haverinnen (Nokia)

Members of the SLD DWG of the VSI Alliance

Context

Reuse of Intellectual Property (IP), or Virtual Components (VCs), from different internal and external sources in Systems-on-Chip, allows companies to focus the R&D to their own core competencies, and to effectively use other companies' specialized expertise for other parts. Such a model can only work if there the microelectronics system industry worldwide can establish an unified vision with a set of open technical standards. This view is quite similar to design practices at the board level today. However, the complexities of future systems-on-chips will largely exceed the ones that we currently know at a board. Moreover, prototypes require costly silicon runs, less signals are visible for probing, less debugging facilities are available, and it will be much more difficult to analyze possible problems when combining several components. Therefore, these virtual components need specific models, to analyse, compare, debug and validate complete system chips and all their interfaces before processing the real silicon, but already starting in the early design phases. This is what is meant today with 'Virtual Prototyping'.

Virtual prototyping of complete hardware (HW) - software (SW) systems is really key, but need to be raised to much higher levels of abstraction than today's design practices, which are usually at the level of synthesizable RTL for custom hardware or Instruction Set Simulator (ISS) for programmable core processors. This shift will result in totally new system level design environments to capture requirements, to specify functionality and architectures, to explore different mappings and schedulings, to select and encapsulate reusable Virtual Components. To be used at 'system level', Virtual Components require several abstract models, expressing e.g. the performance, functionality or cycle-true behaviour.

It is exactly the goal of the System Level Design (SLD) Development Working Group (DWG) of the VSI Alliance to specify standard interfaces, standard data formats and standard methods that will help system designers to explore, debug and verify complex system architectures consisting of several Virtual Components by virtual prototyping at multiple levels of abstraction.

Discussion topics

The discussion should be organized around the main achievements of the SLD DWG of VSIA. These include following areas in the domain of HW/SW co-design and Virtual Prototyping:

Standard nomenclature and VC model taxonomy: Progress to accelerate the encapsulation of Virtual Components (VC) in co-design has hit roadblocks because of a wide diversity of model terminology in use among VC providers, VC integrators, designers, semiconductor companies, system houses and EDA companies. First experiences of integrating third-party components in HW/SW systems by several system companies have shown that different terminology has already created a lot of confusion among the participants. Some organizations use many common modeling terms with divergent meanings, while others use different words to describe the same type of models. While this confusion persists, and the electronics community lacks a common language, different teams will be unable to effectively communicate and share models. Therefore, the SLD DWG has undertaken an effort to develop a nomenclature and modeling taxonomy, which will become a common language to describe models and their attributes for the VSI membership and the electronics design community at large. It contains a classification of

system -, architectural -, hardware - and software models, and is publicly available as standard reference document. It basically modified and augmented previously defined terminology sets, broadened parochial definitions, distinguished overlapping definitions, equated close synonyms, removed inapplicable terms, added new terms, clarified poorly defined or misunderstood ones, and suggested new wording as replacements or synonyms to outdated ones. When appropriate existing definitions were lacking, the SLD DWG created them. Further evolution in design practices for VC integration into System Chips will identify the critical model types. Minimizing the number of models will reduce the effort required to produce a complete design package for a VC.

Standard Interface Behaviour Description: The design methodology promoted by the SLD DWG is based on a clear separation of the VC functionality and the VC interface. Therefore, the DWG is establishing a well-defined hierarchical and multi-level standard description of VC interfaces, covering all abstractions from high level transactions down to detailed timed component protocols, implemented in hardware and/or software. A clean separation of interface properties from VC functionality and behavior, and a clear linking of interface abstractions at the system level, is a significant step towards the achievement of such a goal. A technique for specifying such interfaces will improve VC understanding and utilization. It will reduce the time required to understand behaviors and interfaces correctly. Gaining a faster understanding of VC operational principles allows a system architect to explore many more options before committing to the design phase. This faster VC interpretation and model-integration gives more comfort in the exploration of non-legacy architectures and so will open co-designs to the third-party market. Furthermore, a complete definition of the interface abstraction hierarchy allows designers, architects and SW authors to work within their preferred area of expertise (e.g. embedded-software, RTL, etc.), but still gives them the ability to effectively communicate with the different levels (e.g. unified test-benches/test-results can be applied to any view of the design.). In addition, the standard Interface Behavior description will improve VC model supply and generation, VC integration in HW/SW co-design, and last but not least, VC protection.

Standards for Behavioural Modeling of Virtual Prototypes: For the descriptions of the VC functionality itself, the SLD DWG is shooting for a standard library of data types that are commonly used in behavioral models for virtual prototyping (e.g. in C or C++). Today, system companies are using multiple libraries, even within the same company. Third party VC vendors are offering models with specific libraries, not compliant for other environments. Different syntax and/or semantics make true exchange of models impossible. When high-level models used in virtual prototyping of HW/SW co-design systems can apply all the same data type libraries, the interoperability of these models will be largely enhanced, and co-design will become much more user-friendly.

Performance Modeling Standard for HW/SW systems: This exploration began as part of a recognition that systems containing VCs require high level modeling techniques in order to efficiently evaluate the system performance of interconnected VCs (microprocessors, DSPs, memories, caches, buses, RTOS, etc). The specific intent of this specification is to describe the basic functional and interface requirements of system-level performance models for the most common types of Virtual Components. Performance Models are often the most abstract models of Virtual Components that are used during system design. They describe the system task as well as the resources together with the abstract and physical communication channels. Each of these elements can be modeled in terms of its basic processing and communication capabilities, such as e.g. the rate of processing, the latency of operations, etc. In contrast to functional models, abstract performance models do not compute the results of the operations. System-Level Performance models are used to explore different alternative mappings of the system tasks on selected resource architectures, and to perform trade-off analysis among different hardware and software architectures. It allows the system designer to obtain a first measurement of the quality of the design, to check if the proposed architecture will satisfy the overall performance requirements and meet the constraints, and to identify possible bottlenecks or over-sized parts.